View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarWorks@UMass Amherst

University of Massachusetts Amherst
ScholarWorks@UMass Amherst

Masters Theses 1911 - February 2014

2012

A Study of the Impact of Computational Delays in Missile Interception Systems

Ye Xu
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/theses


Part of the Computer and Systems Architecture Commons, Digital Communications and Networking Commons, and the VLSI and Circuits, Embedded and Hardware Systems Commons

Xu, Ye, "A Study of the Impact of Computational Delays in Missile Interception Systems" (2012). *Masters Theses 1911 - February 2014*. 814.
Retrieved from https://scholarworks.umass.edu/theses/814

This thesis is brought to you for free and open access by ScholarWorks@UMass Amherst. It has been accepted for inclusion in Masters Theses 1911 - February 2014 by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

A STUDY OF THE IMPACT OF COMPUTATIONAL DELAYS IN MISSILE INTERCEPTION SYSTEMS

A Thesis Presented

by

YE XU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

MAY 2012

Department of Electrical and Computer Engineering

© Copyright by YE XU 2012

All Rights Reserved

A STUDY OF THE IMPACT OF COMPUTATIONAL DELAYS IN MISSILE INTERCEPTION SYSTEMS

A Thesis Presented

By

YE XU

Approved as to style and content by:

_____

Israel Koren, Co-Chair

_____

C. Mani Krishna, Co-Chair

_____

Douglas P. Looze, Member

_____

Christopher V. Hollot, Department Head as Department of Electrical and Computer Engineering

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor Prof. Israel Koren and Prof. C. Mani Krishna for their invaluable guidance, many helpful discussions at each research group meeting and attentive reading of many drafts.

I would like to thank Professors Israel Koren, C. Mani Krishna, and Douglas P. Looze for serving in my committee. I thank them for their thoughtful suggestions and comments through my preparation of the proposal to the final defense and revision of this thesis.

Finally, I also would like to thank Dr. Zahava Koren for her help during my research at UMass, and thank my colleagues and friends in my lab (ARTS lab) for their helpful discussions in research and life.

ABSTRACT

A STUDY OF THE IMPACT OF COMPUTATIONAL DELAYS IN MISSILE INTERCEPTION SYSTEMS

MAY 2012

YE XU

B.S, NANJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

M.S.E.C.E, UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Israel Koren, Professor C.M. Krishna

Most publications discussing missile interception systems assume a zero computer response time. This thesis studies the impact of computer response time on single-missile single-target and multiple-missile multiple-target systems. Simulation results for the final miss distance as the computer response time increases are presented. A simple online cooperative adjustment model for multiple-missile multiple-target system is created for the purpose of studying the computer delay effect.

Keywords: Computer response time, computational delay, single-missile single-target system, multiple-missile multiple-target system

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER1

INTRODUCTION

1.1 Objective

The ability of a missile to intercept a target is determined by two factors. The first one is the guidance law employed in the missile's guidance processing and the second one is the flight control system of the air frame. Since a missile system is a cyber-physical system, the embedded computers in it play a fundamental role in determining the performance of the system. The objective of this research is to analyze the effects of certain computing issues on the performance of the missile system, such as the response time incurred in computing the required command signals, noise disturbance during sensing and computing, sampling time period choice, etc.

1.2 Related Work and Literature Review

Embedded systems are real time systems, meaning that computational delays impact the performance as seen by the application. Computational delay (CD or response time) refers to the time elapsed from the point the control algorithm is triggered to the point that the final control command signal is generated. The impact of computational delay on cyber physical systems, or control systems, has been the focus of many research efforts.

In [1], Shin and Cui pointed out that the computational delay can be classified into two categories: (1) computing delay smaller than the sampling time period, and (2) computing delay greater than the sampling time period. The former is referred to as the delay problem, while the latter is called the loss problem, i.e., the loss of the control output. Due to the randomness of the computational delay, which differs from the system time delay (determined by system time constant, also called as "the inertia delay"), it is difficult to characterize this randomness before putting the control system into use. In other words, it is difficult to determine delay in the feedback control loop during the design of the control system. This randomness is caused by the fact that the execution time for a control

algorithm is determined by the outcome of conditional branches, resource sharing delays, and processing exceptions, all of which are data-dependent. So, an upper bound for the computational delay that the system can tolerate is used. The computing time delay is a random variable, and its effects may change throughout the entire lifetime. Different control systems have different structures, thus requiring a separate analysis for each control system. These reasons indicate that in order to accurately analyze the effects of computational delay, we need a case by case study.

Nilsson, et al. [2] further divided computer delays into three parts: Communication delay between the sensor and the controller τsc; Computational delay in the controller τc; Communication delay between the controller and the actuator τca., they studied the randomly varying computational delay in a linear control system, and proposed a Linear-Quadratic-Gaussian (LQG) optimal control to compensate for the computational delay. They also suggested some future work: Optimal schemes when the time delays are correlated from sample to sample and control strategies when the control delay can be larger than the sampling interval. The latter was briefly discussed in [1].

In [3], Wang and Longoria studied the effect of computational delay on the performance of a Hybrid Adaptive Cruise Control System. They described the computational delay as a random number within the range of one sampling time period. It is uniformly distributed on the interval [0, λh], where λ is a bounded value from 0 to 1.0, and h is the sampling time period. The effect of computational delay was roughly analyzed, and then simulation results were compared with the mathematical expressions to confirm the correctness of both methods. A bounded computational delay allowed in the control system was also derived. Since the effect of computational delay is highly dependent on the specific platform, the results from this study cannot be directly used in our missile navigation study, but the ideas and the methods can be used. In this study, a simple lead compensation was employed to compensate for the delay effect.

In [4], Nesline and Zarchan analyzed the tradeoffs between Stability and Performance in a digital homing guidance system. The concept of relative stability was employed to analyze the miss distance.

Open-loop transfer function, gain margin and phase margin, which are all from the classic feedback control theory, were used in this study. Results showed that the miss distance increases as the stability margin degrades. Since the phase margin corresponds to a delay in the time domain, we can use it to determine the maximum delay for system stability. Nesline and Zarchan also showed that to ensure the system stability, each of the system design parameters is dependent on the values of all other parameters. So, in order to make sure the designed system is balanced and robust, not only computational delay effects, but also other parameters, such as system time constants and system gain selections should be examined.

1.3 Thesis Organization

We propose several different methods to analyze certain computer issues in the missile homing loop system. The problem scenario is a two-dimensional intercept geometry of point mass target and aerodynamically influenced missile air frame.

Chapter 2 provides the basic background including the widely used guidance laws, and the fundamental aerodynamic equations for a typical aircraft system.

Chapter 3 presents the implementation of a non-linear model for the missile homing loop system. This model is based on the aerodynamic equations and guidance laws, and several other conditions such as noise level, missile acceleration saturation, and so on. This chapter also contains simulation results of certain performance criteria, such as miss distance and achieved missile accelerations.

Chapter 4 proposes a linearized model, and compares the simulation results with its non-linear counterparts. This chapter also briefly discusses the disadvantages and advantages of both models.

Chapter 5 describes the mathematical functions of one of the most important criteria in the missile homing loop system, namely, miss distance due to computer issues. These mathematical functions focus on the embedded computer systems in the navigation guidance sub systems, ignoring

the embedded computer systems in the flight control sub systems, which would be discussed in the next chapter.

Chapter 6 demonstrates the impact of computational delay on the flight control systems. Here, we mainly concentrate on the problem of stability, since stability is the premise of almost all other issues.

In Chapter 7, a simple but still effective model for an online adjustment algorithm is described. This chapter presents simulation results for the multiple-missile multiple-target system, focusing on two aspects: the effectiveness and advantages over the single-missile single-target system, and its ability to handle the computational delay during flight time.

Finally, chapter 8 gives conclusions and future work.

CHAPTER 2

BASIC BACKGROUND

The aerodynamic equations of a missile are the same as those of an aircraft. Since the purpose of the missile guidance system is to “hit” the target, it needs a specific subsystem, namely, a guidance system. The following two sections briefly summarize the fundamental aerodynamic equations for a typical aircraft system, and the concept of certain classic guidance laws.

2.1 Missile Guidance Laws

This section describes a typical guidance law that is implemented in the missile guidance system, namely, the Proportional Navigation Guidance (PNG).

Proportional Navigation Guidance (PNG) is one of the most widely used guidance laws in homing air target missile systems. The main principle is that if two vehicles are on a collision course, their direct Line-of-Sight (LOS) does not change direction or value. Generally speaking, PNG indicates that the missile velocity direction should rotate at a rate proportional to the turn rate of the LOS, and should be in the same direction.

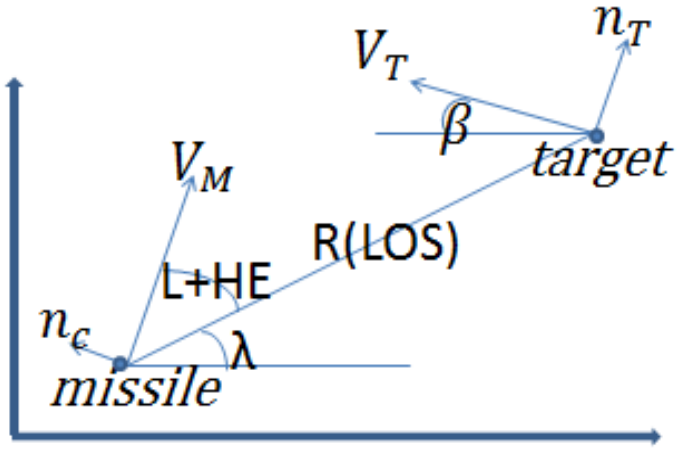


Figure 2.1: Missile-Target Intercept Geometry

Typical parameters in this geometry are:

L+HE: where L is the Missile Lead Angle and HE is the Heading Error. If the missile is heading toward the proper leading angle (see Section 3.A), no further acceleration command is required. Any initial angular deviation of the missile from the proper leading angle is known as the initial heading error, or HE.

R: Length of the line of sight (LOS)

n_C: Missile Acceleration

V_M: Missile Velocity

λ: Angle of Line of Sight (LOS)

β: Target Heading

V_T: Target Velocity

n_T: Target Maneuver (i.e., target acceleration normal to its velocity, which means the target tries to escape by turning.)

Mathematically, the guidance law can be stated as [5][6]:

$$|n_C| = N'|V_c|\dot{\lambda} \tag{2-1}$$

where n_C is the missile acceleration command signal coming from the guidance system (in meter/s²), N' is a dimension-less designer chosen parameter (usually in the range of 3-5) known as the effective navigation ratio, and V_c is the missile-target closing velocity (in meter/s), i.e., the relative velocity between missile and target. |X| is the magnitude of the vector X.

There are more advanced guidance laws that can yield smaller miss distances against a highly maneuvering target, where the miss distance is measured at the point of closest approach of the missile and target. One of these advanced guidance laws is the Augmented Proportional Navigation (APNG) [5][6]:

$$|n_C| = N'|V_c|\dot{\lambda} + N'|n_T|/2 \tag{2-2}$$

APNG is a proportional navigation with an extra term to account for the maneuvering target.

Before we move on to the next step, it is necessary to define the concept of miss distance,

namely, the point of closest approach of the missile and target. Since the purpose of the missile is to hit the target, the desired miss distance would be zero. But due to various factors, in some cases, the miss distance can be high enough so that the missile would fail to enter into the explosive region. In these cases, the missile would completely miss the target. Thus, we need the miss distance to be as small as possible, while keeping other issues such as the total missile acceleration/control effort, in a reasonable range.

2.2 Missile Flight Control System

This section briefly explains the missile airframe representation with transfer functions, including the aerodynamic equations and the flight control system time constants. The details about how these equations are derived can be found in [6].

2.2.1 Aerodynamics and Missile Air Frame Transfer functions [6]

We provide in this section the equations for the aerodynamics and the transfer functions for the missile airframe. The normal force equation can be expressed as:

$$F_N = QS_{ref}C_N \tag{2-3}$$

where CN is the normal force coefficient, Q is the dynamic pressure, and Sref is the reference area. Q and Sref are given by:

$$Q = 0.5\rho V_M^2 \tag{2-4}$$

$$S_{ref} = \frac{\pi d^2}{4} \tag{2-5}$$

where ρ is the air density d and is the missile cross section area diameter.

The normal force coefficient can be approximated as:

$$C_N = 2\alpha + \frac{1.5S_{paln}\alpha^2}{S_{ref}} + \frac{8S_w\alpha}{\beta S_{ref}} + \frac{8S_T(\alpha+\delta)}{\beta S_{ref}} \tag{2-6}$$

where α is the angle of attack, δ is the control surface deflection, and SW, ST and SPLAN are panel wing, tail, and platform areas, respectively. β is a normalized speed and for supersonic travel is

given by:

$$\beta = \sqrt{\mathrm{Mach}^2 - 1} \tag{2-7}$$

The total moment M in the missile can be expressed as:

$$M = QS_{ref}dC_M \tag{2-8}$$

where CM can be approximated as:

$$C_M = \frac{2\alpha(X_{CG}-X_{CPN})}{d} + \frac{\frac{1.5S_{PLAN}\alpha^2}{S_{ref}}(X_{CG}-X_{CPB})}{d} + \frac{\frac{8S_W\alpha}{\beta S_{ref}}(X_{CG}-X_{CPW})}{d} + \frac{\frac{8S_T(\alpha+\delta)}{\beta S_{ref}}(X_{CG}-X_{HL})}{d} \tag{2-9}$$

where XCG is the distance from the nose to the missile center of gravity and XCPN, XCPB, XCPW are the distances from the nose to the center of pressure for the nose, body and wing, respectively. XHL is the distance from the nose to the missile hinge line.

The acceleration normal to the missile body can be expressed in terms of the normal force according to:

$$|n_B| = \frac{F_N g}{W} = \frac{gQS_{ref}C_N}{W} \tag{2-10}$$

where W is the missile weight. The angular acceleration acting on the missile can be expressed in terms of the moment according to:

$$\ddot{\theta} = \frac{M}{I_{yy}} = \frac{QS_{ref}dC_M}{I_{yy}} \tag{2-11}$$

where Iyy is the missile moment of inertia, and can be approximated as:

$$I_{yy} = \frac{W[3(0.5d)^2+L^2]}{12g} \approx \frac{WL^2}{12g} \tag{2-12}$$

The transfer function relating the achieved missile normal acceleration to the fin deflection (fin is the control surface of the missile, fin deflection means how much deflection angle of the control surface should be made) can be expressed as:

$$\frac{|n_L|}{\delta} = \frac{K1\left(1-\frac{S^2}{\omega_Z^2}\right)}{1+\frac{2\zeta_{AF}}{\omega_{AF}}S+\frac{S^2}{\omega_{AF}^2}} \tag{2-13}$$

Where:

$$K1 = -\frac{V_M[M_\alpha Z_\delta - Z_\alpha M_\delta]}{1845M_\alpha} \tag{2-14}$$

And:

$$\omega_z = (M_\alpha Z_\delta - Z_\alpha M_\delta)/Z_\delta \tag{2-15}$$

$$\omega_{AF} = \sqrt{-M_\alpha} \tag{2-16}$$

$$\zeta_{AF} = \frac{Z_\alpha \omega_{AF}}{2M_\alpha} \tag{2-17}$$

The transfer function from missile pitch rate to fin deflection can be written as:

$$\frac{\dot{\theta}}{\delta} = K3(1 + T_\alpha s)/(1 + \frac{2\zeta_{AF}}{\omega_{AF}} s + \frac{s^2}{\omega_{AF}^2}) \tag{2-18}$$

Where:

$$K3 = -\frac{[M_\alpha Z_\delta - Z_\alpha M_\delta]}{M_\alpha} = \frac{1845K1}{V_M} \tag{2-19}$$

$$T_\alpha = M_\delta/(M_\alpha Z_\delta - Z_\alpha M_\delta) \tag{2-20}$$

2.2.2 Homing Loop and the Flight Control Systems

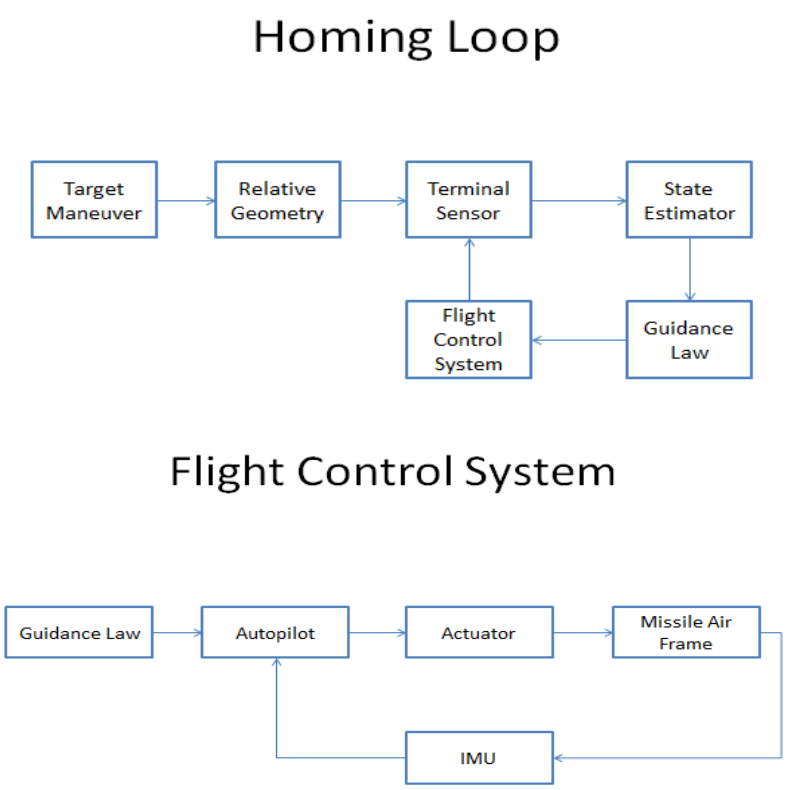


Figure 2.2: Missile Homing Loop and Flight Control System [10]

The Missile Homing Loop system includes the flight control system. The relative geometry is a combination of the inertial missile motion controlled by the flight control system and the target motion and specifically the target maneuvering. The function of the terminal sensor is to measure the missile-to-target LOS rate. Then, the output of the terminal sensor is forwarded to the state estimator, which generates the estimation of the LOS angle rate, which in turn provides the input to the guidance law. The output of the guidance law is the command signal, and is typically a translational acceleration required for the missile to change its direction. The flight control system takes advantage of the missile control surface such as the control tail to force the missile to move according to the command signal.

The flight control system consists of mainly four parts, namely, the autopilot, actuator, missile air frame, and the inertial measurement unit (IMU). The IMU measures the inertial motion of the missile,

and combines it with the guidance command signal together as the input to the autopilot. The autopilot then generates a control signal, such as the fin deflection, to the aerodynamic control surface. The actuator turns the electric control signal generated by the autopilot into the physical control effort (i.e., how much fin deflection should be made by the actuator), which in turn influences the missile airframe to track the guidance command signal.

The Missile Homing Loop is a feedback system that regulates the LOS rate to zero. The Flight Control System is another feedback loop within the homing loop that receives the guidance command generated by the guidance system.

CHAPTER 3

NON-LINEAR IMPLEMENTATION OF MISSILE HOMING LOOP

The non-linear implementation is accurate for simulation, but too complex for analysis and generation of a mathematical expression for the miss distance. For example, the closed form solution for PNG is obtained directly from the linearized missile-target engagement model. In the following several sections we first implement a non-linear missile homing loop based on the fundamental guidance laws and aerodynamics and missile airframe transfer functions discussed above. Then, we provide a linear approximation for the Homing Loop System, and show that both non-linear and linear systems generate almost the same simulation results. Then, we include the additional response time for computation in the embedded processors in the Homing Loop System and analyze the miss distance due to the computational delay.

3.1 Missile Leading Angle Calculation

From figure 2.1, the leading angle (L) means that if the missile is on its leading angle, no further acceleration commands are required for the missile to hit the target if both missile and target keep their initial trajectory.

According to the Law of Sine:

$$\frac{|V_T|}{\sin L} = \frac{|V_M|}{\sin(\lambda+\beta)} \tag{3-1}$$

Solving for L we get:

$$L = \sin^{-1}\frac{|V_T|\sin((\lambda+\beta))}{|V_M|} \tag{3-2}$$

All of the Missile initial conditions are generated in the launch computer (which means the leading angle is unique).

3.2 MATLAB Simulink Block Diagram of Missile Homing Loop

Below are some variables and notes about the nonlinear model:

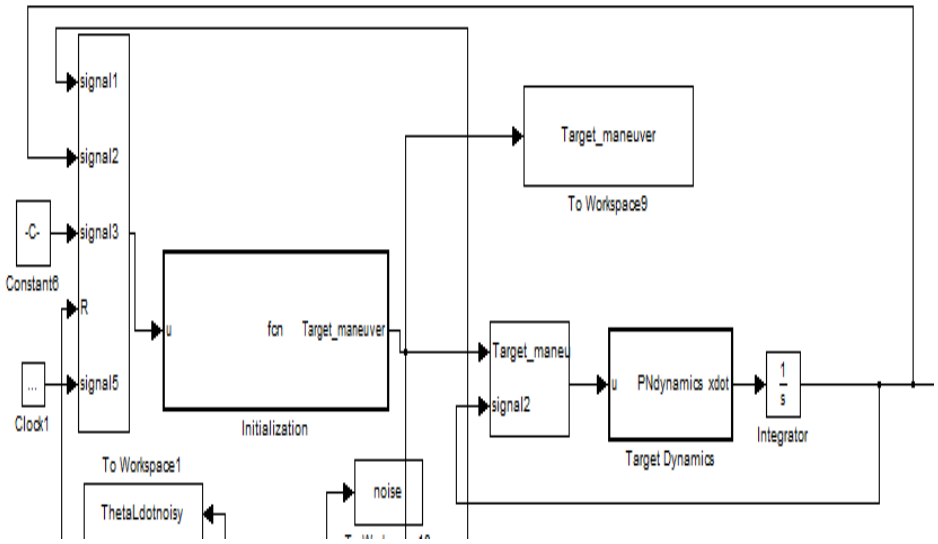


Figure 3.1: Initialization Block and Target Dynamics Block (The bottom missing part is actually figure 3.3)

Initialization Block: It is used to initialize the sensing range of the target, and the target maneuver. Its main inputs are the instant relative range, the value of the target maneuver after the missile enters the target sensing range, and one clock recording the instant time of the simulation. Its output is the target maneuver, which is 0 before the missile enters the sensing range and the predefined value after entering the range.

Target Dynamics block: its inputs are the target maneuver and the instant target position and velocity along x and y axis. We define the values in MATLAB as follows:

U (1) = target maneuver;

U (2) = target x-axis location

U (3) = target x-axis velocity

U (4) = target y-axis location

U (5) = target y-axis velocity

The main code inside the block is based on the state vector equation of the target as follows: The brackets for the u vector should be as tall as those for the matrix:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -u(1) \\ 0 & 0 & 0 & 1 \\ 0 & u(1) & 0 & 0 \end{bmatrix} \begin{bmatrix} u(2) \\ u(3) \\ u(4) \\ u(5) \end{bmatrix}$$

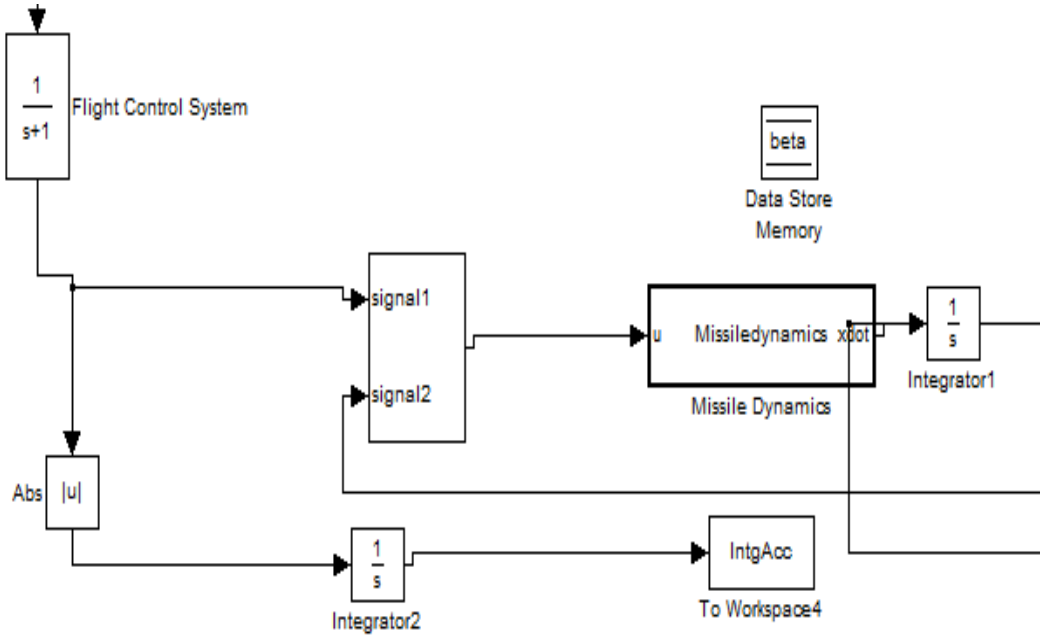


Figure 3.2: Missile Dynamics Block - it is similar to the target dynamic block, except that it includes additional aerodynamic equations to calculate the extra velocity lost during the flight

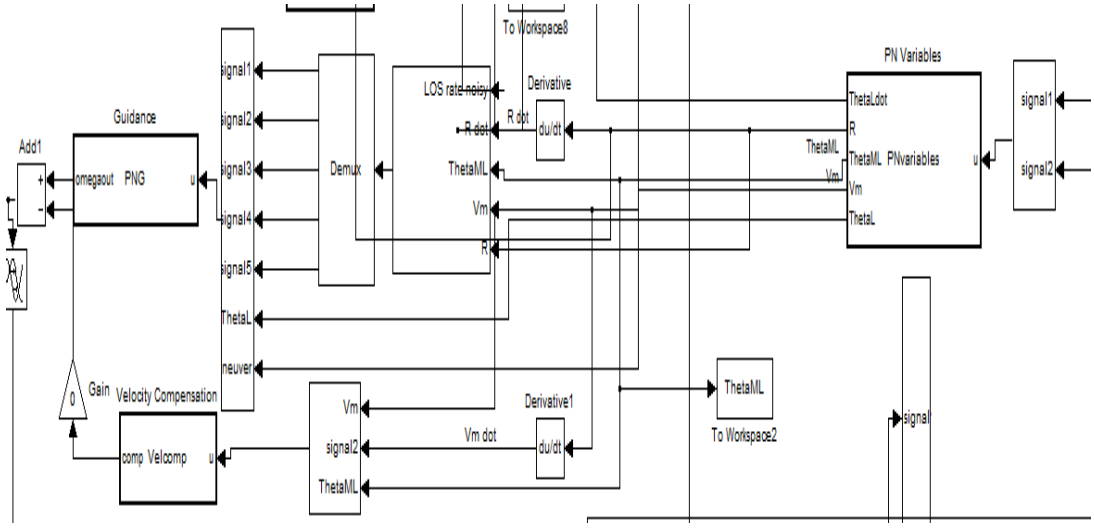


Figure 3.3: PN Variables Block and PNG Block (the top missing part is Figure 3.1)

PN Variables Block: it calculates all the variables needed for PNG, which are the LOS rate, and the closing velocity, which can be calculated here by $\dot{R}$, the derivative of the instantaneous relative range.

PNG Block: this is the PNG guidance block and it calculates the command signal based on PNG.

3.3 Simulation Results Using Non-Linear Implementation

Below are a few simulation results based on the following assumptions:

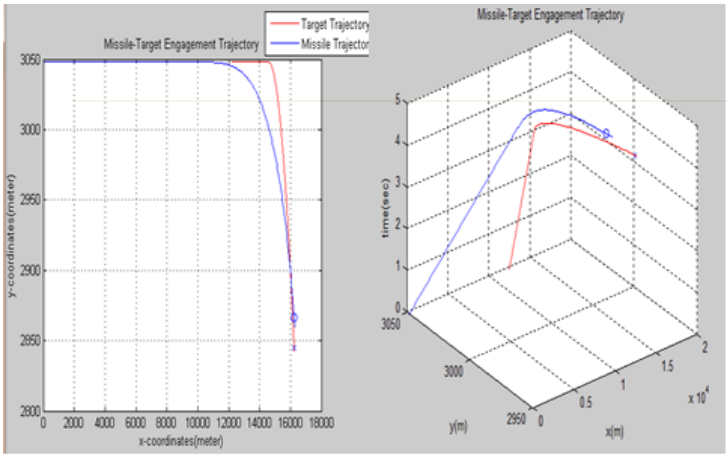
No Noise

Target is a mass point with a first order transfer function $\frac{1}{0.5s+1}$ [6]; Missile is a rigid body, influenced by aerodynamic forces (has velocity reduction); and Missile Airframe is 1st order system, time constant of 1, with a transfer function: $\frac{1}{s+1}$ [6]

Target sensing range: 3000m (once the relative distance between the target and missile is less than 3000m, the target starts to escape)

Target maneuvering: 10g [6]; Missile maneuvering saturation: 3 times as much as target maneuvering.

Initial Target velocity: 1000m/s [6]; Missile Initial Velocity: 4000 m/s [6] (ballistic missiles).



Computer Response Time: 0.01s .
Miss Distance : 24 m.

3D Version for the Case of 0.01s Response Time (with additional time-axis)

Figure 3.4: A Simple Case for Response Time of 0.01s

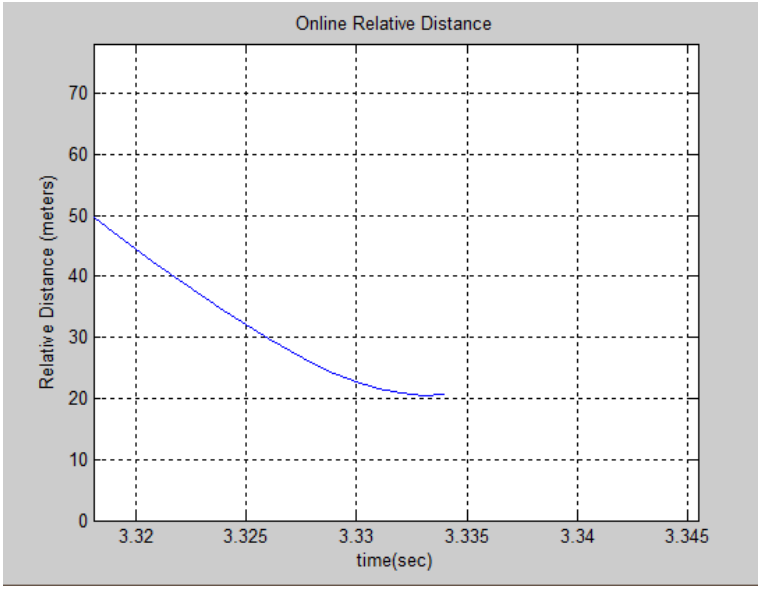
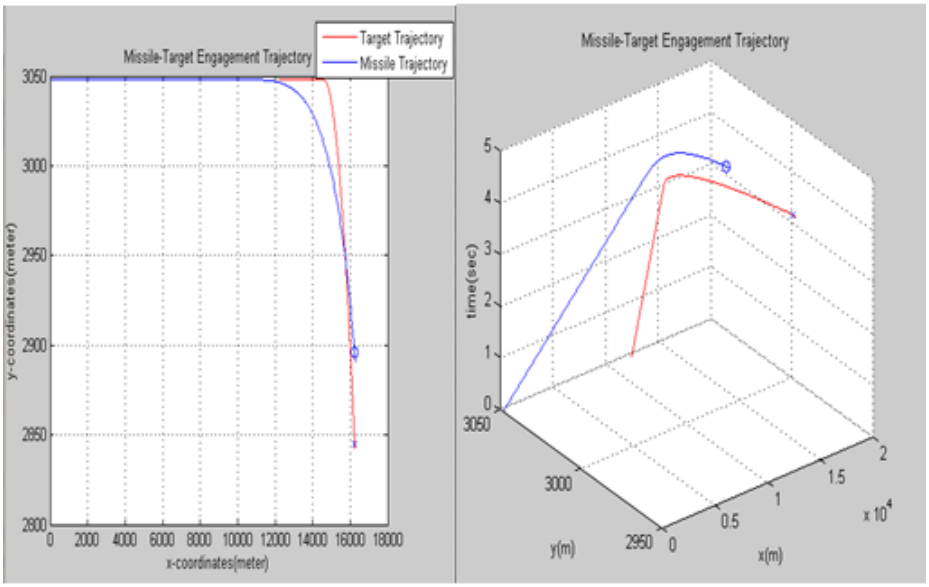


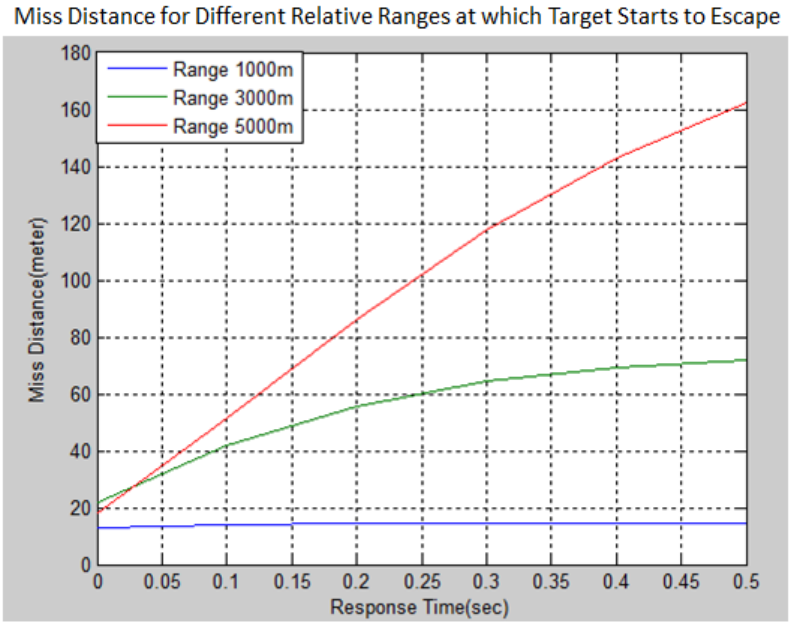
Figure 3.5: Online Relative Distance between Missile and Target around the time Target Starts to Escape



Computer Response Time: 0.1s .
Miss Distance : 42 m.

3D Version for the Case of 0.1s Response Time (with additional time-axis)

Figure 3.6: A Simple Case for Response Time of 0.1s



Note: Here we set the computer response time up to 0.5 seconds, only for the purpose of analysis. In reality, it cannot be such a high delay.

Figure 3.7: Miss Distance for Different Relative Ranges at which Target Starts to Escape

CHAPTER 4

LINEARIZATION

4.1 Assumptions to Simplify the Model

Thus far, our results are based on the performance of PNG in a two dimensional non-linear situation. For simplicity, we would prefer a linear model. In this section, we indicate how closely the nonlinear system can be linearized.

From Fig. 2.1, we have:

$$\Delta y'' = |n_T| \cos\beta - |n_C| \cos\lambda \tag{4-1}$$

and normally LOS is small, so:

$$\lambda \approx \frac{y}{R},\ y'' \approx |n_T| - |n_C|\ ,|v_c| = |v_M| - |v_T| R = |v_c|(t_F - t) \tag{4-2}$$

where:

$$t_F = (R|_{t=0})/(|Vc||_{t=0})$$

is the total flight time. Also we have:

$$\text{Miss} = y\ (t_F)$$

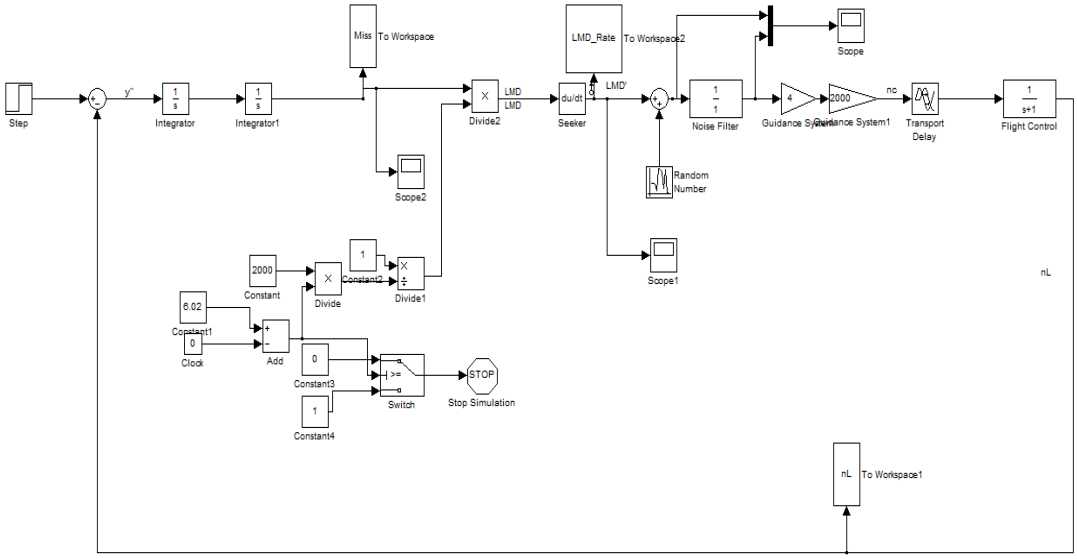
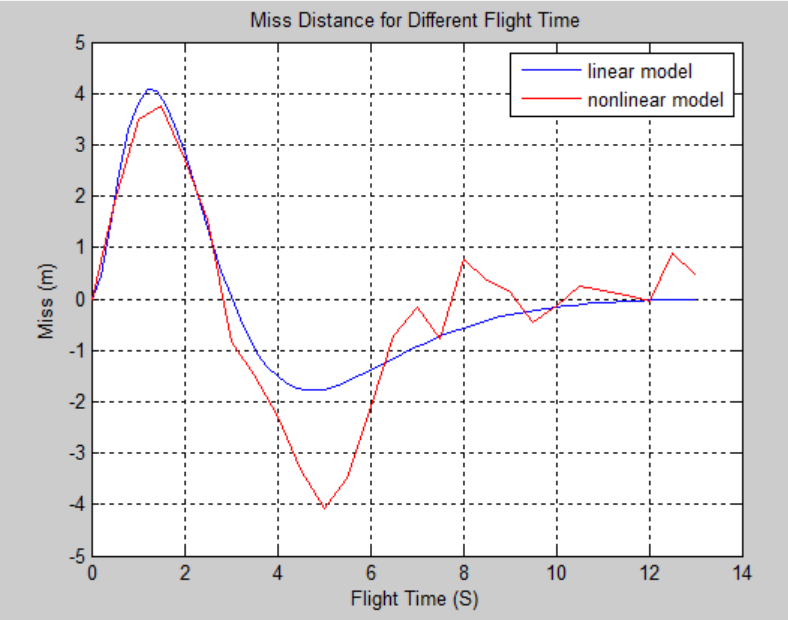


Figure 4.1: MATLAB SIMULINK block diagram of the linear model. Here the flight control system is a first order system.

4.2 Comparison between Non-Linear and Linear Model



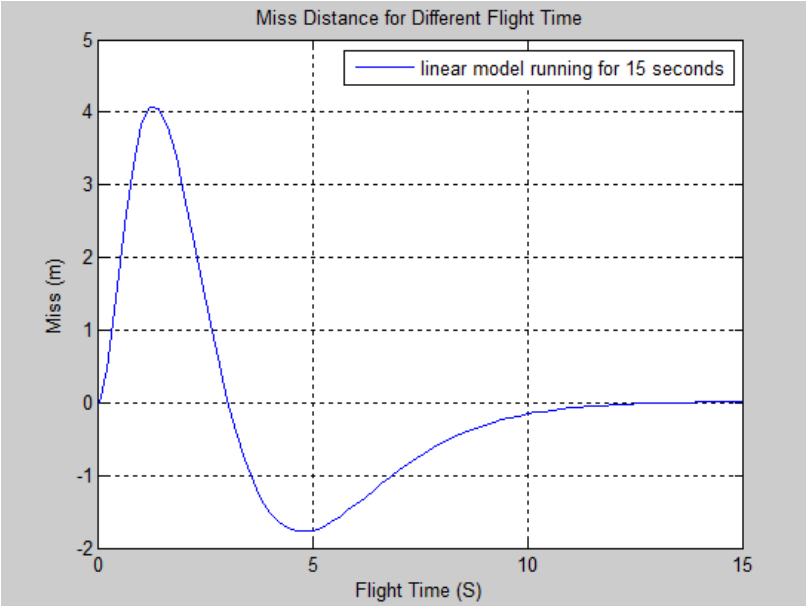


Figure 4.2: Comparison between the linear and the non-linear models: Target maneuver is 3g, Missile 1st order system, Time Constant of 1s. (Note that Flight time means time to go when the target starts to maneuver, similar to the relative range when the target starts to escape)

From Figure 4.3 we can see that, first of all, the simulation results of the linear model are close to those of the nonlinear one, especially the trend of miss distance as flight time increases. The negative value means that the missile is below the target at the end of simulation. It is clear that as the flight time increases, the miss distance first increases, then decreases, then increases but as negative values, then decreases again. When the flight time is greater than 8 seconds, the miss distance is "settling down", i.e., is close to 0.

Additionally, the non-linear model is considerably more complex than the linear one. Typically it takes more than 10 seconds for each run of the nonlinear model while for the linear model, it can run more than 70 times during the same time.

Thus, although the nonlinear model may be more accurate, we will use the linear model for most of the simulations and analysis. The nonlinear model can be used to verify the results from the linear model. If they are close, we can conclude that the results are correct.

4.3 Simulation Results for the Linear Model

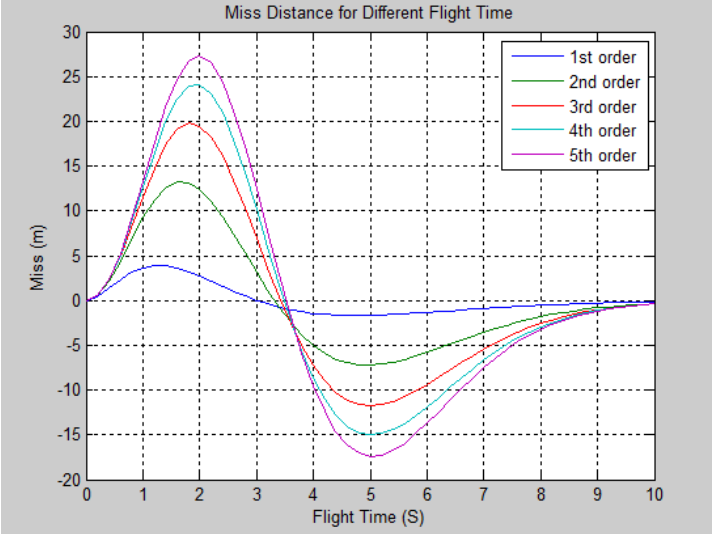


Figure 4.4: Different curves of Miss Distance for different flight times. Target has 3g maneuver; Missile is a 1st – 5th order system, and the Time Constant is 1s.

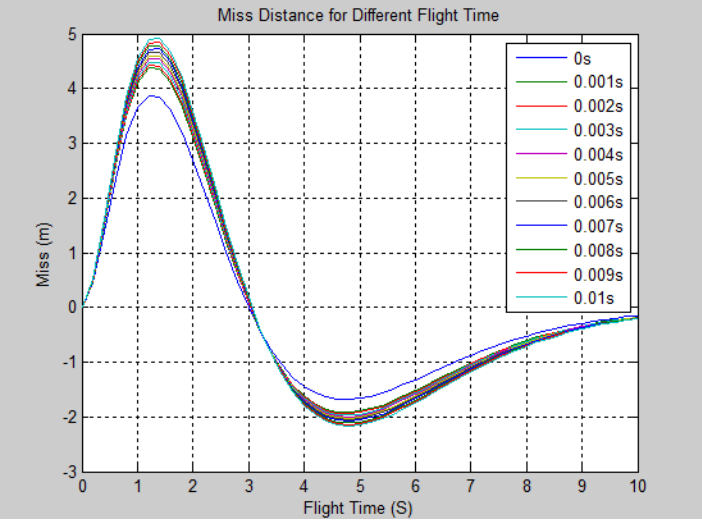


Figure 4.5: Different curves of Miss Distance for different flight times. Target has 3g Maneuver, Missile is 1st order system, Time Constant is 1s. Delay is 0.001s-0.01s

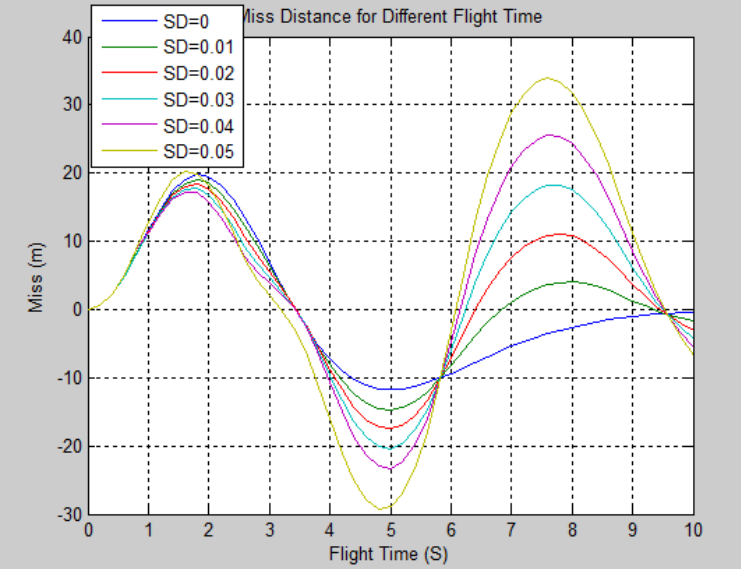


Figure 4.6: Different curves of Miss Distance for different flight times. Target has 3g maneuver, Missile is a 3rd order system, Time Constant is 1s. Delay=0s, Noise Level: SD (Standard Deviation) =0-0.05

CHAPTER 5

MATHEMATICAL FUNCTIONS OF MISS DISTANCE DUE TO DELAY

5.1 Closed Form Solution for PNG (Based On the Linear Model)

Below is the closed form solution for PNG [6]:

$$|n_c| = \frac{N'}{N'-2}\left(1-\left(1-\frac{t}{t_F}\right)^{N'-2}\right)|n_T| \tag{5-1}$$

where t_F is the total flight time, which is approximately given by:

$$t_F = (R|_{t=0})/(|Vc||_{t=0} \tag{5-2}$$

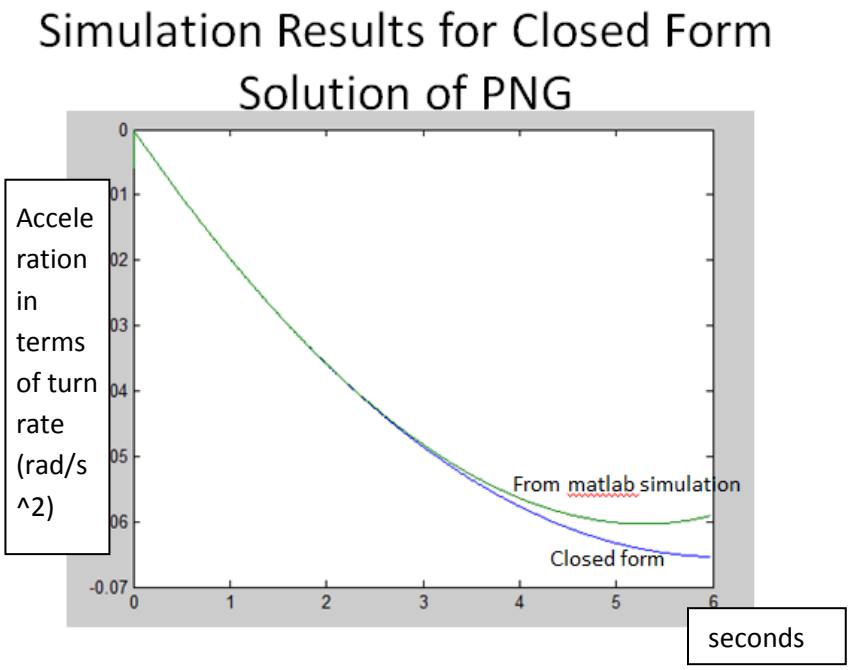


Figure 5.1: Simulation results comparing the closed form solution for PNG and the Non-Linear MATLAB simulation for PNG

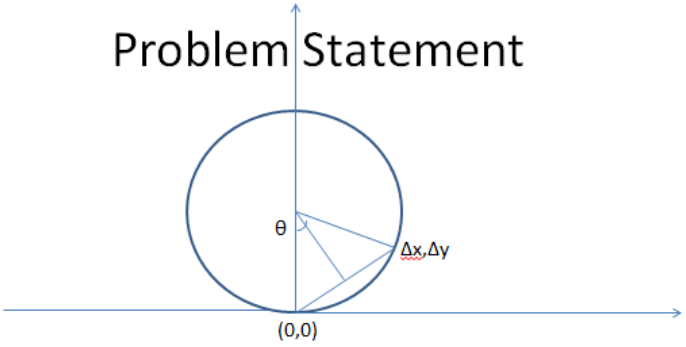
5.2 Equations of Circular Motion Given Normal Acceleration:

Physical Equations for Circular Motion:

Angular Velocity : $\omega = 2\pi/T$

Linear Velocity: $v = \omega R$

Centripetal Acceleration: $a_c = \frac{v^2}{R} = \omega^2 R$

- 

Given a_c, v, and initial position(0,0), we can get the position for any given value of time(t):

$$\Delta x = 2R * \sin\frac{\theta}{2} * \cos\frac{\theta}{2}$$

$$\Delta y = 2R * \sin\frac{\theta}{2} * \sin\frac{\theta}{2}$$

Where:

$$\theta = \frac{t}{T} * 2\pi,\ T = \frac{2\pi R}{v},\ R = v^2/a_c$$

5.3 Closed Form Solution For Target Movement:

At each sampling time period, in the absence of computational delay and dynamics, both missile and target would move in a basic circular motion.

After some derivations, we get expressions for Δx and Δy for the target displacement at each sampling time period:

$$\Delta x = \frac{2v^2}{a_c}\sin\left(\frac{t * a_c}{2v}\right)\cos(\frac{t * a_c}{2v}),$$

$$\Delta y = \frac{2v^2}{a_c}\sin\left(\frac{t * a_c}{2v}\right)\sin(\frac{t * a_c}{2v})$$

5.4 Closed Form Solution For Missile Movement:

There are two cases for the delay:

Case 1. Delay is smaller than the sampling time period. In this case, if the delay time is ε, it means that from t_i to $t_i + \varepsilon$, the missile still has the command acceleration from the last sampling time, say, $n_{c,i-1}$. From $t_i + \varepsilon$ to t_{i+1}, the new command signal is updated, which lets the missile move based on the updated acceleration, say, $n_{c,i}$.

Case 2. Delay is greater than the sampling time. In this case, it means that due to the delay, the missile lost some updated information from the controller causing worse result.

5.5 First Case: Command Signal If Delay Time (ε) is Less than Sampling Period:

At each sampling time, there are two acceleration commands for the missile (total flight time is $t_i = i * T$):

From t_i to $t_{i+\varepsilon}$, due to computational delay, the command signal is still the old one:

$$|n_{c,t_{i-1}}| = \frac{N'}{N'-2}\left[1 - \left(1 - \frac{t_{i-1}}{t_F}\right)^{N'-2}\right]|n_T| \tag{5-3}$$

From $t_{i+\varepsilon}$ to t_{i+1}, the command signal is the updated one:

$$|n_{c,t_i}| = \frac{N'}{N'-2}\left[1 - \left(1 - \frac{t_i}{t_F}\right)^{N'-2}\right]|n_T| \tag{5-4}$$

Missile Movement in each sampling period: Make sure that equations numbers are in the same line as the equation:

$$\Delta x = \int_{t_i}^{t_i+\varepsilon} v_x\, dx + \int_{t_{i+1}+\varepsilon}^{t_{i+1}} v_x dx \tag{5-5}$$

$$\Delta y = \int_{t_i}^{t_i+\varepsilon} v_y\, dy + \int_{t_{i+1}+\varepsilon}^{t_{i+1}} v_y dy \tag{5-6}$$

$$v = v(0) + a(t) * t \tag{5-7}$$

$$a_x(t_i) = a(t_i)\cos(\theta_{previous} + \omega t + \frac{\pi}{2}) \tag{5-8}$$

$$a_y(t_i) = a(t_i)\sin(\theta_{previous} + \omega t + \frac{\pi}{2}) \tag{5-9}$$

where t in (5-7) (5-8) (5-9) is a time variable within one sampling time period (0-0.01s in our case)

$$\mathrm{Miss}_{x,kT} = \mathrm{Tgt}_{\Delta x,kT} - \mathrm{Missile}_{\Delta x,kT} \tag{5-10}$$

$$\mathrm{Miss}_{y,kT} = \mathrm{Tgt}_{\Delta y,kT} - \mathrm{Missil} \tag{5-11}$$

$$\mathrm{FinalMiss}_{x} = \sum_{k=0}^{k=N} \mathrm{Miss}_{x,kT} + \text{Initial distance} \tag{5-12}$$

$$\mathrm{FinalMiss}_{y} = \sum_{k=0}^{k=N} \mathrm{Miss}_{y,kT} + \text{Initial distance} \tag{5-13}$$

5.6 Second Case: Command Signal If Delay Time ($\boldsymbol{\varepsilon}$) Is Longer Than Sampling Period:

If the delay is longer than the sampling time period, the missile loses some updated information from the controller resulting in poor results. In other words, the actual sampling time period now is the delay time, instead of the desired sampling time period, and due to the longer sampling time period, the performance (miss distance) would be worse.

Using the previous mathematical functions, replacing the sampling time period by the delay time, we can get the mathematical expressions in the case of a delay time longer than the sampling time period.

5.7 Impact of Delay:

We can consider the equations with and without delay together, and obtain the performance cost of delay as follows:

$$\Delta X = \left[v_{x,t_{i-1}}(t_i) - v_{x,t_i}(t_i+\varepsilon) - a_{x,t_i}(t_i+\varepsilon)T\right]\varepsilon + \left[\frac{1}{2}a_{x,t_{i-1}}(t_i) + \frac{1}{2}a_{x,t_i}(t_i+\varepsilon)\right]\varepsilon^2 + v_{x,t_i}(t_i+\varepsilon)T + \frac{1}{2}a_{x,t_i}(t_i+\varepsilon)T^2 - v_{x,t_i}(t_i)T - \frac{1}{2}a_{x,t_i}(t_i) \tag{5-14}$$

$$\Delta Y = \left[v_{y,t_{i-1}}(t_i) - v_{y,t_i}(t_i+\varepsilon) - a_{y,t_i}(t_i+\varepsilon)T\right]\varepsilon + \left[\frac{1}{2}a_{y,t_{i-1}}(t_i) + \frac{1}{2}a_{y,t_i}(t_i+\varepsilon)\right]\varepsilon^2 + v_{y,t_i}(t_i+\varepsilon)T + \frac{1}{2}a_{y,t_i}(t_i+\varepsilon)T^2 - v_{y,t_i}(t_i)T - \frac{1}{2}a_{y,t_i}(t_i)T^2 \tag{5-15}$$

Here ΔX and ΔY are the difference between missile displacement without delay and that with delay, in each sampling period. If we want to calculate the pure delay effect on the miss distance, i.e.,

the difference between the miss distance without delay and that with delay, we can simply apply the previous equations to the above two equations and obtain the following (note that target displacements are the same in both cases, that is, in both cases the target displacements are $\mathrm{Tgt}_{\Delta x,kT}$ and $\mathrm{Tgt}_{\Delta y,kT}$):

$$\text{Difference in } \mathrm{Miss}_{x,kT} = \mathrm{Tgt}_{\Delta x,kT} - \mathrm{Tgt}_{\Delta x,kT} + \Delta X = \Delta X \tag{5-16}$$

$$\text{Difference in } \mathrm{Miss}_{y,kT} = \mathrm{Tgt}_{\Delta y,kT} - \mathrm{Tgt}_{\Delta y,kT} + \Delta Y = \Delta Y \tag{5-17}$$

$$\text{Difference in } \mathrm{FinalMiss}_{x} = \sum_{k=0}^{k=N} \Delta X \text{ + Initial distance} \tag{5-18}$$

$$\text{Difference in } \mathrm{FinalMiss}_{y} = \sum_{k=0}^{k=N} \Delta Y \text{ + Initial distance} \tag{5-19}$$

CHAPTER 6

DELAY EFFECT ON THE FLIGHT CONTROL SYSTEM

6.1 Rate Gyro Flight-Control System (A Simple Case)

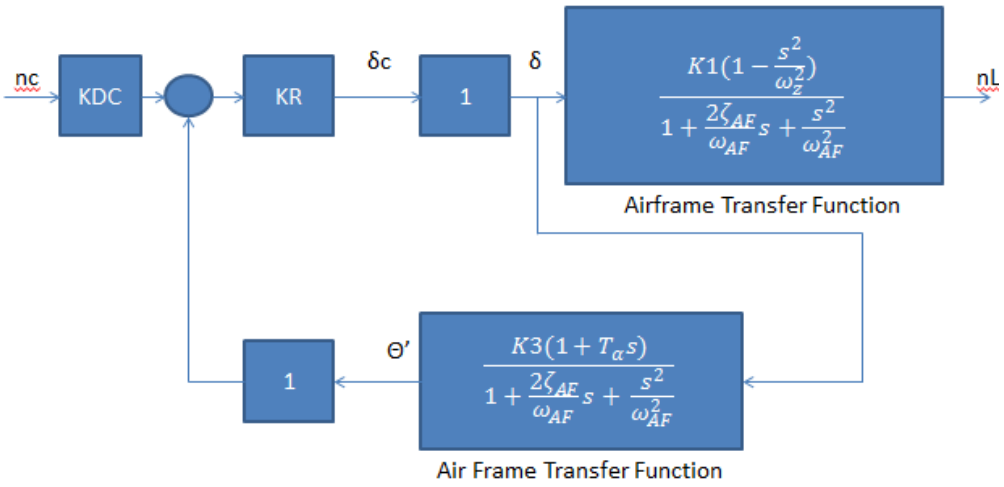


Figure 6.1: RATE GYRO FLIGHT-CONTROL SYSTEM [6]

In Figure 6.1 KDC and KR are two outputs (gains) of the Autopilot controller, KR determines the response of the flight control system, while the gain KDC controls the steady-state value of the response; in other words, KR determines the transient response and stability of the system.

KR changes with different flight attitude (e.g. typical values: at sea level, 0.1; at 12000 meters high, 0.25), and KDC $=\frac{1-K_R K_3}{K_1 K_R}$.

The algorithm for generation of KR and KDC are called gain scheduling.

In this case, the missile air frame is represented as the aerodynamically influenced transfer function, instead of the simple 1st -5th order systems, as shown before. Since we are now concerned with the delay effect on the flight control system itself, it is necessary to take a more realistic missile air frame into account.

Typical Values

Airframe parameter	Definition	Sea Level	12000 meters
wAF	Airframe natural frequency	25.3rad/s	10.0 rad/s
zAF	Airframe damping	0.058	0.027
wz	Airframe zero	43.2 rad/s	18.9 rad/s
K1	Aerodynamic acceleration gain	- 3.07 g/deg	- 0.559 g/deg
Talpha	Turning rate time constant	0.457s	2.40s
K3	Aerodynamic body rate gain	0- 1.89 1/s	0- 0.344 1/s

Figure 6.2: Typical Values for Several Parameters in the Rate Gyro Flight Control System

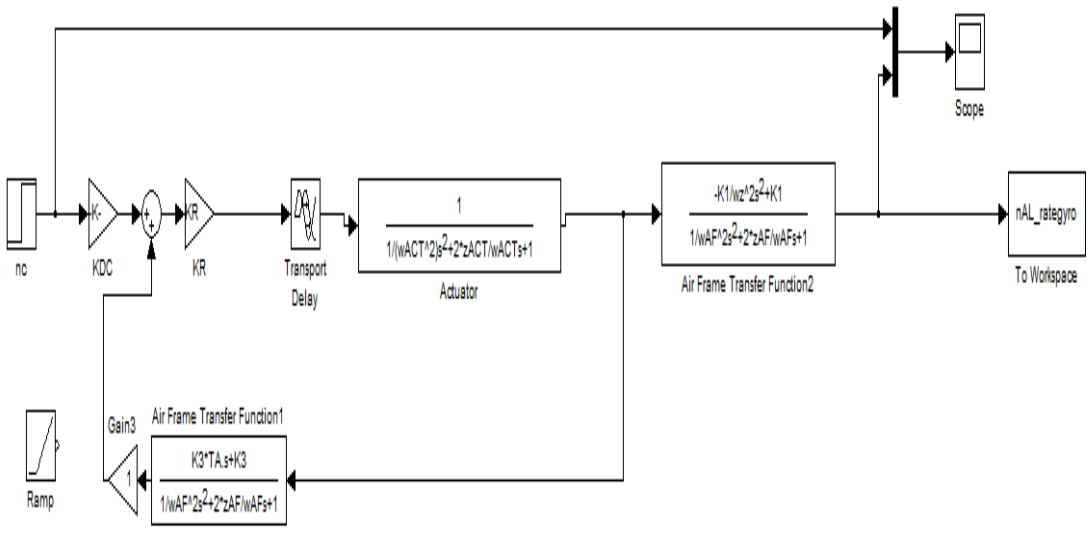


Figure 6.3: MATLAB SIMULINK Block Diagram of Rate Gyro Flight Control System

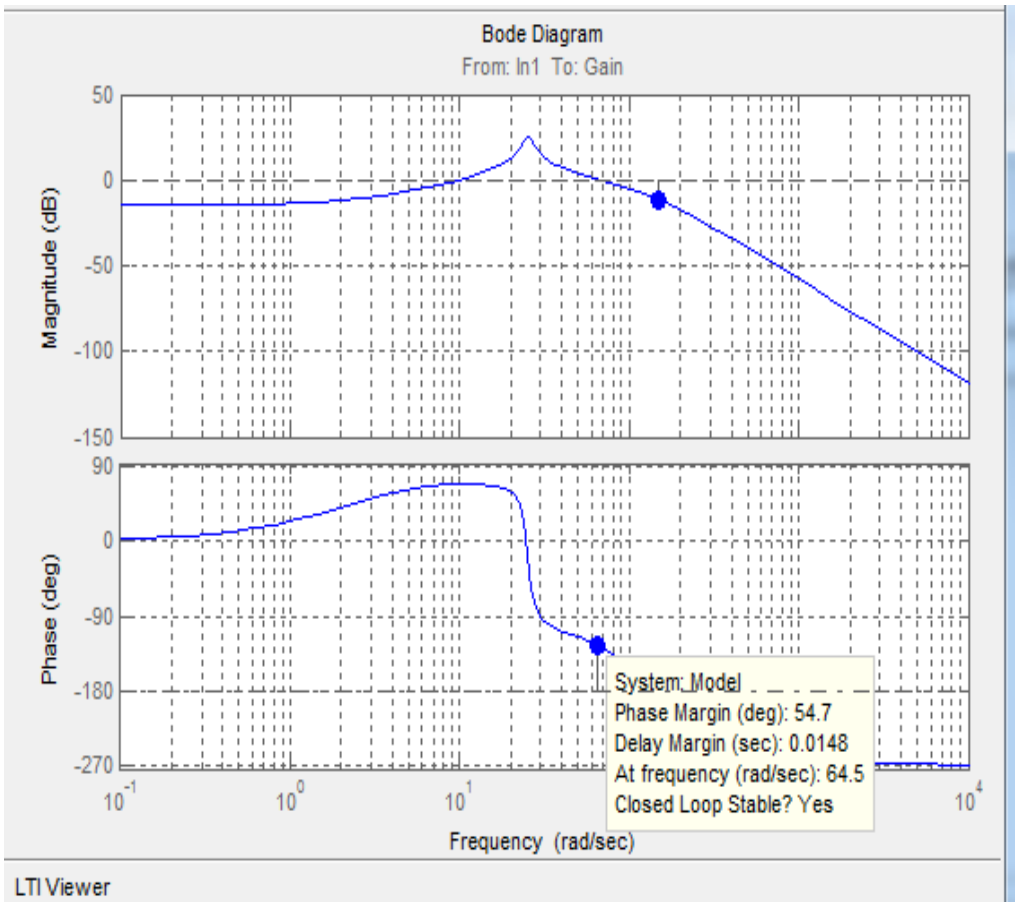


Figure 6.4: Bode Diagram of Open Loop of the Rate Gyro Flight Control System with a breaking point at the Point of KR. From a feedback control point of view, if we want to determine the maximum delay allowed for the system to stay in the stable region, we can use the concept of Phase Margin.

Here we determine the Maximum Delay for the output of KR before the system would become unstable: If the system phase is decreased by the phase margin, the system will become unstable and oscillate at the gain cross over frequency.

The Phase Margin here is about 55 deg; the crossover frequency wCR is 64 rad/s. The maximum delay time can be calculated, as follows (T is the maximum delay time): 55 deg = 55/57.3 rad = 0.96 rad. This means that 64T = 0.96, so T = 0.015s. (Very close to the delay margin of 0.0148 shown in the Bode Diagram).

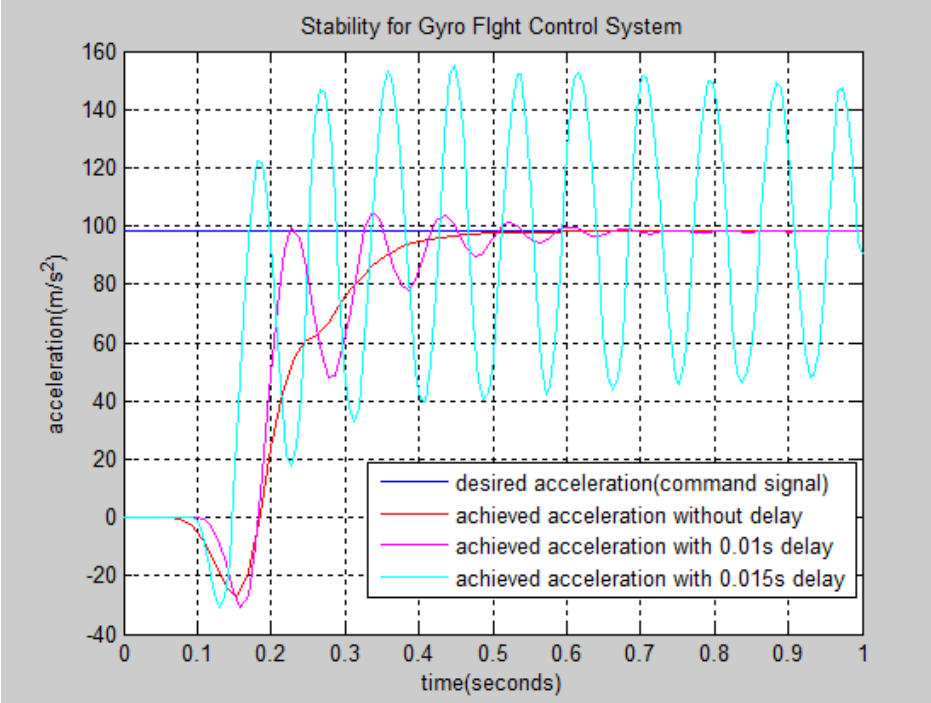


Figure 6.5: Simulation Results for the Delay Effect of KR on the Rate Gyro Flight Control System

6.2 CLASSIC THREE LOOP AUTOPILOT FLIGHT CONTROL SYSTEM:

The same method can be used to get the maximum delay that allows ensuring the system to be stable in the case of a three loop auto pilot control system. This autopilot can control both the damping and the time constant at the same time.

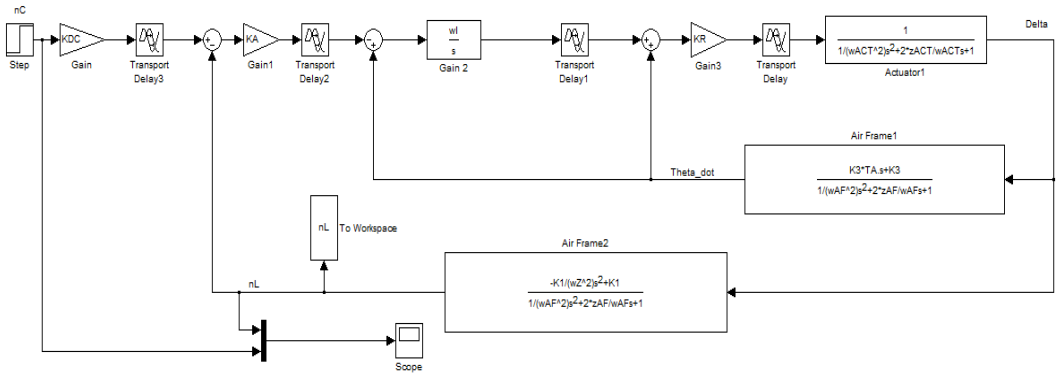


Figure 6.6: MATLAB SIMULINK Block Diagram for a Three Loop Autopilot Flight Control System [6]

KA, wI, and KR are selected in order to meet the design criteria such as stability, natural frequency and damping (i.e., the stability and the transient response) for the entire closed-loop three loop autopilot flight control system, whereas KDC is computed from the other three gains in order to control the steady-state value (error).

If we want to determine the maximum delay allowed for the output of KR that still allows the system to be stable, we can use the following steps:

First, specify the desired system criteria: time constant, natural frequency and damping and cross over frequency.

Second, use the Gain selection algorithm to get the four gains (assume no delay)

Third, get the open loop frequency response for the open loop system breaking at the point of KR (-output(KR)/input(Delta)), draw the Bode Diagram and determine the phase margin, and then determine the delay.

Open loop breaking at the point of KR

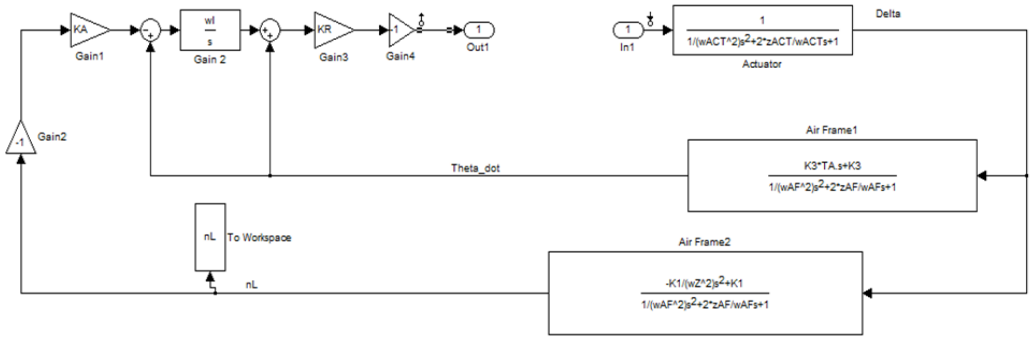


Figure 6.7: Open loop breaking at the point of KR

Bode Diagram for Open Loop Frequency Response breaking at the point of KR

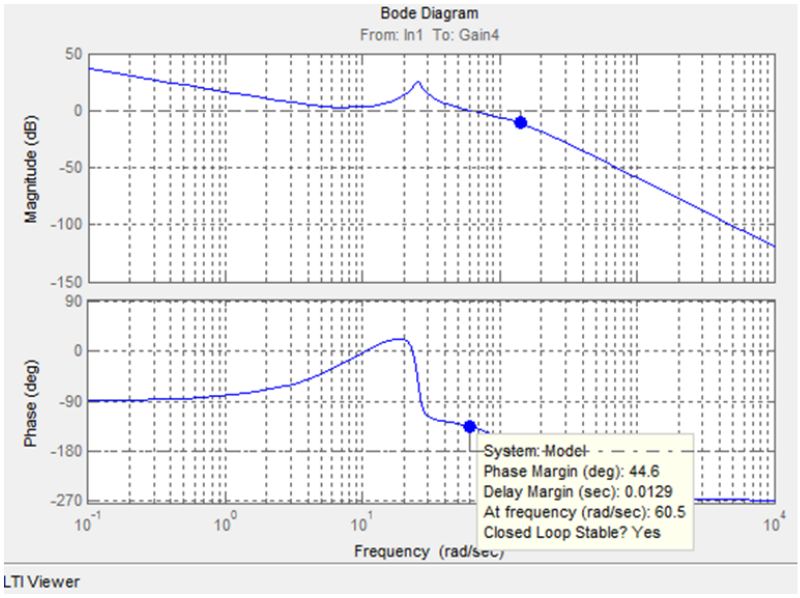


Figure 6.8: Bode Diagram for Open Loop Frequency Response breaking at the point of KR. We can directly get the Delay Margin (maximum delay) from this figure: 0.0129.

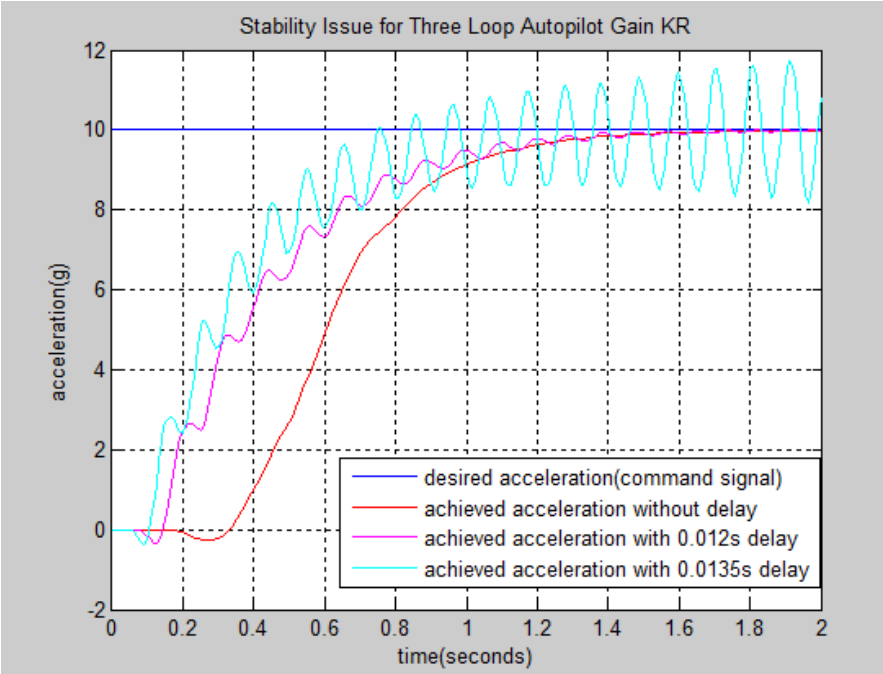


Figure 6.9: Simulation Results for the Delay Effect of KR on a Three Loop Autopilot Flight Control System

Since there are four gain generations, the same method can be used to determine the maximum delay or the delay margin allowed for the system to be stable.

Above are two cases that the desired system performance criteria are specified: time constant, natural frequency and damping and cross over frequency. In a more general case, we should be able to use the original letters (like t, KR, and so on) instead of specific numbers to calculate the maximum allowed delay.

CHAPTER 7

MULTIPLE MISSILE MULTIPLE TARGETS COOPERATIVE SYSTEM

7.1 Introduction

The analysis above is based on a single missile single target system. We reached conclusions regarding what factors would affect the final miss distance. In general, if a more advanced and complicated guidance law is implemented inside the embedded processor, and a more expensive radar system is used, the final performance would be improved.

Traditionally, the missile interception scenario is a one to one situation. But nowadays, high value and high threat targets (like invading missiles) are very hard to anticipate and intercept. An improved single missile system with multi-spectral seeker systems can achieve smaller final miss distance against highly maneuvering target, but at a high cost. Another approach is to launch a multi to one engaging system (a.k.a, the salvo missile attack). These salvo attack missiles can only have one simple seeker for each missile and each missile can estimate the dynamics and position of the target and perform a traditional attack on the target. Salvo missile attack could have the same seeker system for each missile member to get the improved target dynamics and position estimation.

But a salvo attack missile system would have to implement different guidance algorithms in order to achieve different flight profiles (i.e., different flight trajectories). If every missile applies exactly the same guidance law with the same parameters, the interception success would not be considerably improved, since a single mode of elusive maneuvering of the target would be sufficient for escaping from all such antimissiles. Even if applying different parameters for the same guidance law, the missile flight trajectories would only be slightly different.

Moreover, modern and future missile defense systems are also required to consider multiple targets scenario, with different flight profiles for each target. Launching a salvo attack for each invading target would be too expensive and unnecessary. In such situations, some sort of cooperative

missile attack system is generally required to handle modern threat target systems. The basic idea is that by adding communication capability to missile members, data collected by sensors from each target and other missile members can be fed to each missile, allowing each missile to estimate each target dynamics and perform attacks.

Such a cooperative missile system usually needs to be able to adjust missile engagement plan online effectively. This involves tradeoff between computation, accuracy, response time, and so forth. Online real time requirements require that the cooperative algorithm should not be too complicated so that it would not miss many deadlines. Accordingly, due to this online requirement, the engagement should not fall into two extreme cases: the first one is that it is unnecessary and impossible to re-adjust engagement plan at each time step; the second one is that it cannot be only decided at the beginning, which would make the system fail due to the dynamically changing situation.

For the rest of this thesis, a simple enough but still effective online adjustment algorithm is proposed and implemented. It enables the cooperative missile system to change three parameters during the flight time: which target index to engage for each missile members; which guidance law to employ; and which guidance parameters to feed to the guidance law. The guidance laws considered here are the basic proportional navigation and the augmented proportional navigation, since these two algorithms are the simplest and easy to implement.

7.2 Simulink Model

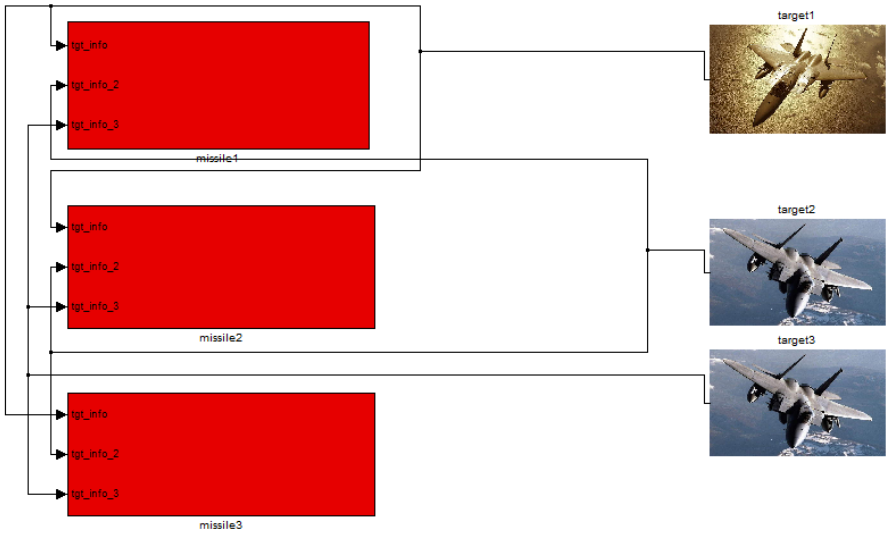


Figure 7.1 Multiple Missile Multiple Targets System

This figure shows the big picture of the multiple missiles multiple targets system. Basically, the targets give output as the target information to each missile member, or in other words, each missile senses each target information as their input to the system. Target information is essentially an information vector including: target index ID, target x and y axis location, target x and y axis velocity.

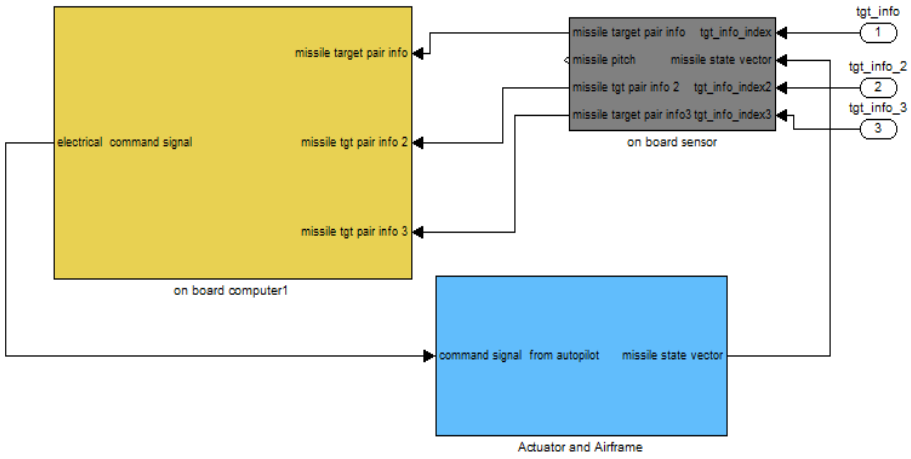


Figure 7.2: Missile Block

This figure shows the structure inside the red missile block. The yellow block represents the missile embedded computer system; grey block stands for the sensor system; and the blue block defines the missile actuator and airframe. The data flow in this figure shows that after each target information vector is sensed by the on board sensor, the sensor system combines the missile's own information vector with the target information vector and feed them into the on board computer system. The output of the computer system is the electrical command signal that indicates how much maneuvering this missile should make for the engagement. And the actuator and missile airframe block takes this command signal as input and generates the actual missile state vector, which is in turn fed to the sensor system to be combined with the targets information.

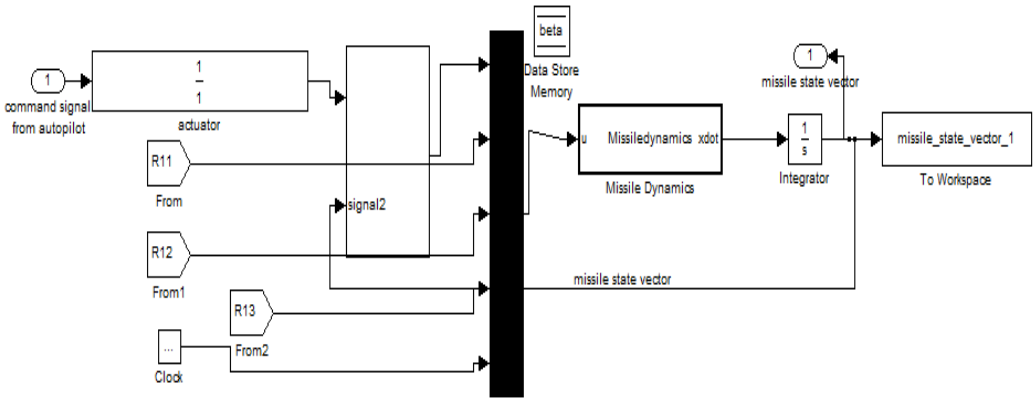


Figure 7.3 Missile and Target Actuator and Body Diagrams

This figure shows the missile and target actuator and body diagrams. For simplicity, in our Online Adjustment Algorithm, we assume the actuator is perfect, i.e. it immediately transfers the electrical command signal to the airframe without delay. The missile or target dynamics block shown above is the same as the single missile engagement system shown previously in this thesis.

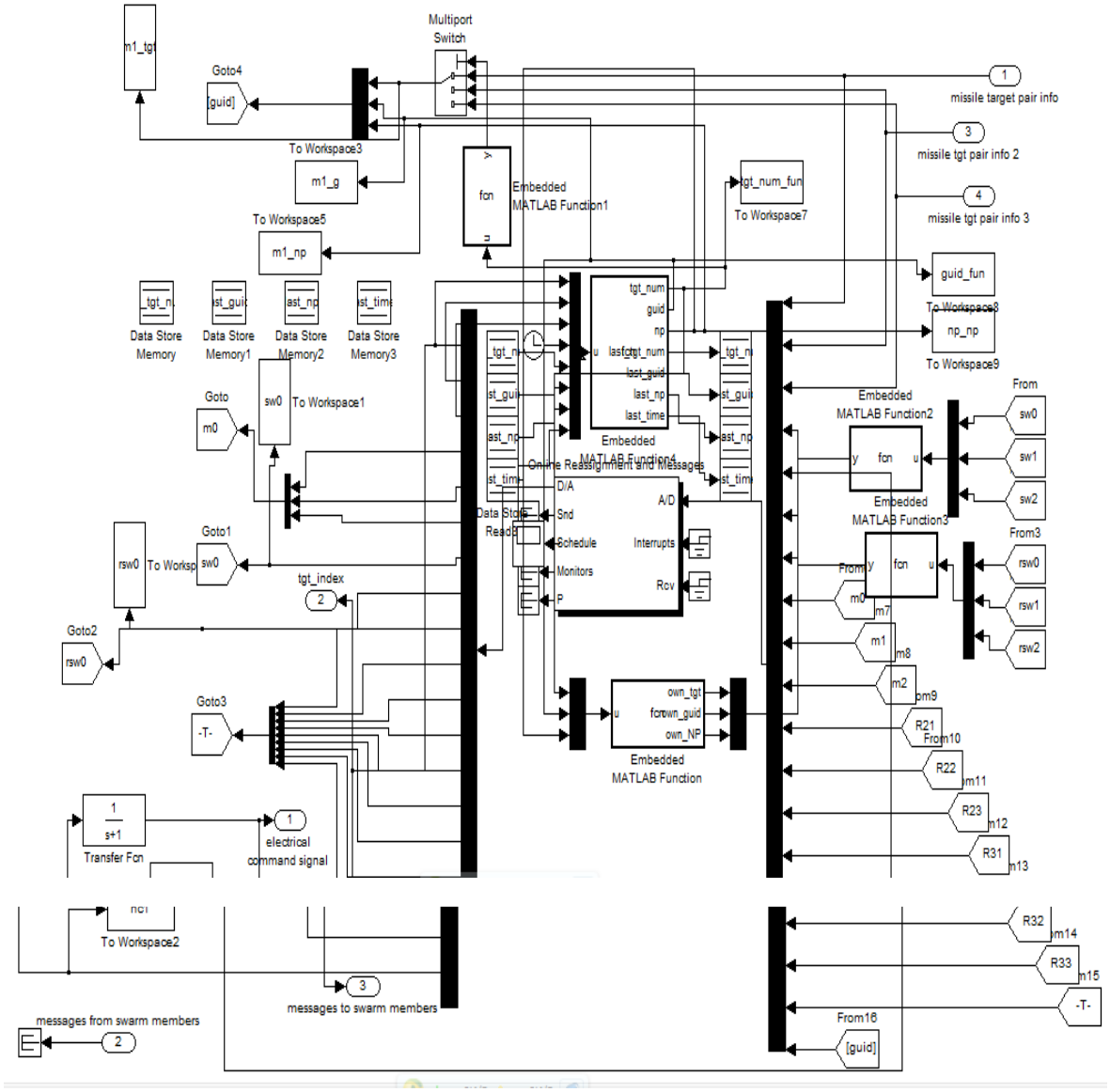


Figure 7.4: Missile Onboard Computer System

This figure above shows the embedded computer system. Basically, all of the data and information, including missile and targets information vectors, messages from other missile members, received go into the truetime real time kernel (designed by real time computing group at Lund University), in which the source code for all the algorithms, including Online Cooperative Adjustment Algorithm, PNG and APNG, live. The output of the computer system is the solution vector sol[] including three elements: the target index this missile should engage, the guidance law this missile

should employ, and the guidance law parameter the missile should feed to the guidance law.

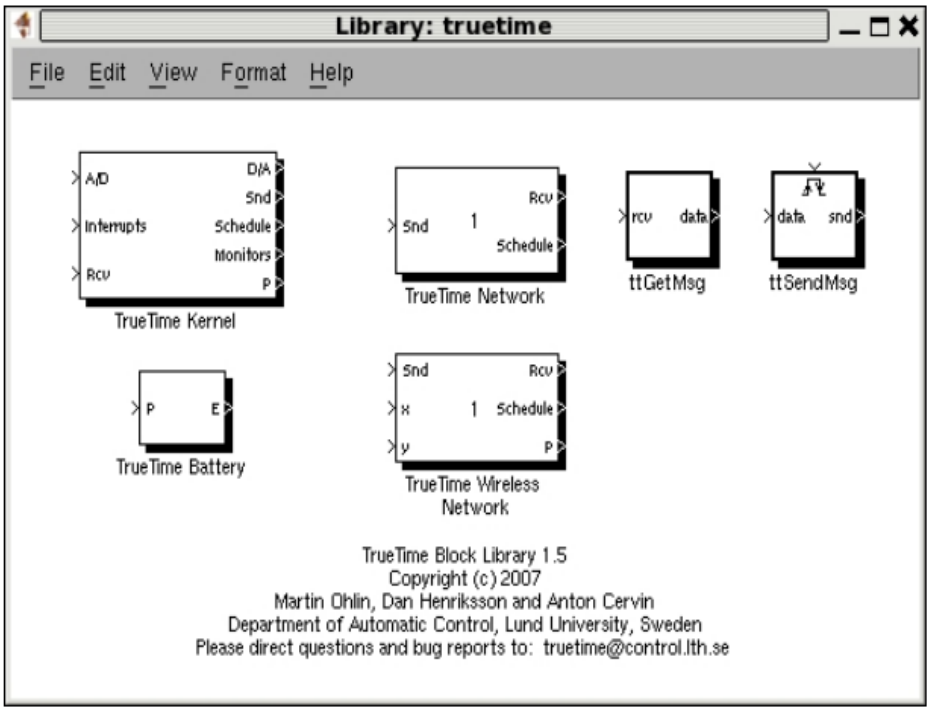


Figure 7.5: True Time Real Time Kernel

7.3 Online Cooperative Adjustment Algorithm

The Online Cooperative Adjustment Algorithm (OCAA) has three sub tasks. The solution vector sol[] includes three elements: the target index this missile should engage, the guidance law this missile should employ, and the guidance law parameter the missile should feed to the guidance law. We are currently using two guidance laws, PNG and APNG, with 0 standing for PNG and 1 for APNG. For example, if the solution vector is sol [1, 0, 3], it means that the current missile is engaging target 1, with PNG and the parameter N' 3. We choose PNG as the default guidance law, since PNG is the simplest and still effective guidance law against non-maneuvering or small maneuvering target. If the target is highly maneuvering so that PNG cannot handle, APNG is an alternative. However, the noise level for the measured target maneuvering level increases when APNG is used. [6]

The first sub task is a periodic task, simply passing the current missile's last solution vector to the missile computer, and evaluating this solution. The on-board computer would check two things at this time. The first one is whether the final estimated miss distance using the last solution vector would be within the missile's explosive range, the second one is to check if other missile members sent a switch target request to the current missile. If both are false, then the current missile continues using the last solution vector without changing anything.

The second sub task is that if during the online evaluation the missile finds out that the last solution vector cannot guarantee a hit, the on-board computer would generate a switch request signal to other missiles. At the same time, the computer would also generate new solution vectors for each missile target pair, and save all the available solution vectors for each target for later use, as well as generate the information vector showing which target the current missile can hit to be sent to the other missiles. For example, if missile 0 finds out it cannot guarantee hitting target 0 anymore during the online evaluation, it would first send a switch request signal to the other missiles. Then, missile 0 would try to generate all the available solutions for each target, for example, if missile 0 could hit target 1 with the solution vector sol[1, 0, 3] which means missile 0 could hit target 1 with PNG and N' 3, then missile 0 would save this vector for later use. At the same time, if missile 0 could finally hit target 1 and 2, it would generate an information vector [0, 1, 1], which means it would miss target 0 and hit target 1 and 2, and would send this information vector to all other missiles.

The third sub task would be that after every missile gets the information vectors from all other missiles, all missiles would combine these information vectors into an information table. Then, each missile's on board computer would make assignment using the same assignment algorithm. This finishes the online adjustment and the missile computer would return back to sub task one.

Below are the pseudo codes for this Online Cooperative Adjustment Algorithm (OCAA):

FOR each time period

SENSE this missile's and each target's flight condition: [x-position, x-velocity, y-position, y-velocity];

IF this missile's solution [target ID to engage, guidance law ID to employ, guidance parameter to feed] from last period predicts the final miss distance to be within the explosive range ***AND*** no other missile member sent switch-request signal = 1 ***THEN***

Use the same solution for this missile;

Send switch-request signal = 0 to other missiles;

Send ready-to-switch signal = 0 to other missiles;

ELSE IF this missile's solution [target ID to engage, guidance law ID to employ, guidance parameter to feed] from last period predicts the final miss distance to be outside the explosive range ***THEN***

Use the same solution for this missile;

Send switch-request signal = 1 to other missiles;

Send ready-to-switch signal = 0 to other missiles;

ELSE IF at least one of other missiles has sent switch-request signal = 1 ***AND*** ready-to-switch signal from all missiles are 0 ***THEN***

FOR each target ID

FOR each guidance law ID (here we use two guidance laws, PNG and APNG, so ID should be 0 and 1, respectively)

FOR each guidance law parameter N' (here we use 3,4,5)

Generate trial_sol[target ID, guidance law ID, guidance parameter N'];

Get predicted final miss distance for each trial_sol[], if estimated final miss distance miss_for_trial[targetID] is within the explosive range, set this as the missile-target pair's proposed_sol[targetID][], **break the loop**

END FOR

END FOR

END FOR

Send miss_for_trial[] to other missiles;

Send proposed_sol[][] to itself for later use;

Send switch-request signal = 1 to other missiles;

Send ready-to-switch signal = 1 to other missiles;

ELSE IF at least one of other missiles has sent switch-request signal = 1 ***AND*** all other missiles have sent ready-to-switch signal = 1***THEN***

Combine all miss_for_trial[] from other missiles and itself to a matrix miss_info[][];

FOR each element in miss_info[][]

Set as 1 if smaller than the explosive range, 0 otherwise, and put these 0s and 1s in a new table info[][];

END FOR

Initialize assignment array sol[](each element is the target ID that missile ID is assigned to, e.g., sol[0] = 1 means missile 0 is assigned to target 1.), each element set as -1;

Assign();(this function is to make assignment based on the miss_info[][] table, after this is done, sol[] is generated. This algorithm uses a simple backtrack and forward checking)

```
solution[0] = proposed_sol[sol[this_missileID]][0];

solution[1] = proposed_sol[sol[this_missileID]][1];

solution[2] = proposed_sol[sol[this_missileID]][2];
```

Send switch-request signal = 0 to other missiles;

Send ready-to-switch signal = 0 to other missiles;

END IF

END FOR

7.4 Estimation and Evaluation

In the algorithm above, an important part is the evaluation of each trial solution, estimating the final miss distance and determining whether the final miss distance is within the missile explosive range. The trial solution can be used as the proposed solution for this missile and sent to other missiles.

Each missile runs a simulation for each time step and based on the outcome of the simulation, the objective, which is the predicted final miss distance for each missile-target pair, is

calculated. The simulated flight will start at the current missile position and end at an estimated impact point that is predicted from the current target position and trajectory; unfortunately the impact point is not known exactly until the missile reaches it, therefore requiring a simulation. The classic proportional navigation guidance law and the augmented proportional navigation law (depending on which law is being used at that time step) are used to determine the instantaneous positions required to estimate the final miss distance, and then to guide the missile onto the moving target.

The approach does mean that the current state of the engagement can be taken into account; however, the process is slow if a very high fidelity simulation is to be performed. Therefore, the simulation of each trajectory is performed using very coarse approximations of the flight dynamics to reduce the processing time, but the coarse approximations do however introduce uncertainties which this online adjustment algorithm can effectively handle. Below is the pseudo code for the evaluation:

FOR each time period

SENSE the current missile and target information to the missile computer, including missile position and velocity **[mx,my,mvx,mvy]**, target position velocity **[tx, ty, tvx,tvy]** and estimated acceleration **acclt**;
PASS current missile guidance law **guid** and guidance law parameters **N'** to the missile computer;

INITIALIZAE: sampling time **h** = 0.002; **vm** = sqrt(mvx^2 + mvy ^2), **vt** = sqrt(tvx ^ 2 + tvy ^2); target maneuver **nt** = wt * vt; **initial miss distance** (i.e., current missile target relative range) = sqrt((mx-tx)^ 2)+(my-ty)^2); estimated total flight time **tf** = initial miss distance / initial relative velocity; **miss_old** = initial miss distance;

FOR (int i = 0; i < tf; i = i + h)
IF miss ≤ miss_old, which means that the missile has not missed the target yet
ASSIGN current miss to miss_old;
ASSIGN current missile and target information [mx,my,mvx,mvy] and [tx, ty, tvx,tvy] and acclt to old ones;
CALCULATE current missile and target heading **m_heading** and **t_heading**; line of sight **LOS**; line of sight rate **LOS_dot**; relative range and velocity etc;

CALCULATE current miss distance using current guidance law and guidance law parameters based on classic **Runge-kutta method** for the calculation of

achieved missile acceleration **nl** from determined missile acceleration **nc**;

ELSE

CLACULATE the **estimated final miss distance** using current missile and target information [mx,my,mvx,mvy] and [tx, ty, tvx,tvy];

RETURN estimated final miss distance;

END IF

END FOR

END FOR

7.5 Runge Kutta Method

Since the command signal for missile acceleration nc cannot be achieved immediately by the missile body, the command signal nc is fed to the missile autopilot which moves the missile control surface accordingly. The missile autopilot and missile airframe control surface can be represented as transfer functions based on some linearization. Thus, the achieved missile acceleration nl applied on the missile control surface can be achieved approximately based on the classic Runge-kutta method, or 4th order Runge-Kutta method, since transfer functions are essentially differential equations.

Simply put, we assume the transfer function between nc and nl, i.e., the missile autopilot and the missile airframe is the first order transfer function (one can always generate higher order transfer functions to perform more accurate simulation results, but since this evaluation algorithm should have small response time, we choose a first order transfer function for the approximation). Let the initial problem be specified as follows:

$$y' = f(t, y); y(t_0) = y_0; \tag{7-1}$$

The 4th order Runge-Kutta method gives the following equations:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4); \tag{7-2}$$

$t_{n+1} = t_n + h$; h is the sampling time period where y_{n+1} is the same thing as $y(t_{n+1})$

and,

$$k_1 = hf(t_n, y_n); \tag{7-3}$$

$$k_2 = hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right); \tag{7-4}$$

$$k_3 = hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right); \quad (7\text{-}5)$$

$$k_4 = hf(t_n + h, y_n + k_3); \quad (7\text{-}6)$$

This Runge-Kutta method is easy to program, and most importantly, yields accurate answers for all the problems in this work. For details about this Runge-Kutta method, one may refer to Runge-Kutta method on Wikipedia.

7.6 Back Tracking Assignment

Once we have the information table from all of the missiles that indicates whether each missile target pair is a hit or a miss, we can make the missile to target assignment. This is a typical linear assignment problem of how to assign n targets to n missiles in the best possible way. In this case, if we have an information table below:

$$\begin{matrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

The row number represents the missile ID, and the column number represents the target ID. So in this case, the entry in row 0 and column 0 is 0 which means that missile 0 is likely to miss target 0, whereas the entry in row 0 and column 2 is 1 which means missile 0 has a good chance of hitting target 2. We also can notice that missile 2 is unable to hit any target in this particular case, since row 2 has three 0 elements.

The solution of the linear assignment problem using backtracking can be found in any algorithm textbook, the basic idea of the algorithm is that: starting from the element in the first row and first column, iterate using either row-based or column-based order, assign the first available 1 as the proposed missile target pair, then use forward checking to delete the row and column of this element, since we cannot assign the same missile to different targets, or assign two different missiles to the same target at the same time. Keep going until we have a final solution, or we get stuck in the middle of the process because using the current approach there is no solution satisfying all missile target

pairs. If the latter, we backtrack to the last row or column, depending on whether it is row-based or column-based iteration, and then choose the next available missile target pair that has an element 1, and keep going again. We continue this backtracking and forward checking approach until we find the final solution, or we have tried every possible combination but it is impossible to find a successful engagement for each missile target pair. For instance, in the above information table, missile 2 is unable to hit any target, which means at least one target would be missed. However, even if this situation happens, we still have achieved the best possible way that allows as many missiles as possible to successfully hit the targets.

As an example, we will look at a more complicated case, which has 6 missiles engaging 6 targets. The information table is as follows:

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{matrix}$$

$$\begin{matrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{matrix}$$

Suppose we are using a column-based iteration, which means we are making assignment for each target. We look at the first column, and assign missile 1 (Row 2) to target 0, since this is the first element 1 for this column (i.e., target 0). We then check the second column, namely, target 1, and assign missile 0 to target 1 in the same manner. We continue, and assign missile 3 to target 2, and then assign missile 2 to target 3. However, when we move to column 4, target 4, we find that the only available missile for this target is 2, which has already been assigned. So we backtrack to the last column, and find that other than missile 2, there is no available missile left for target 3, since by forward checking, we already deleted rows 1 and 3. Therefore, we again backtrack to the last column, namely, column 2, and assign the next available missile, missile 4, to target 2. Then we assign missile 2 to target 3 and again find that we are stuck with target 4, so we backtrack and assign missile 3 to target 3, and assign missile 2 to target 4. Finally, we assign missile 5 to target 5.

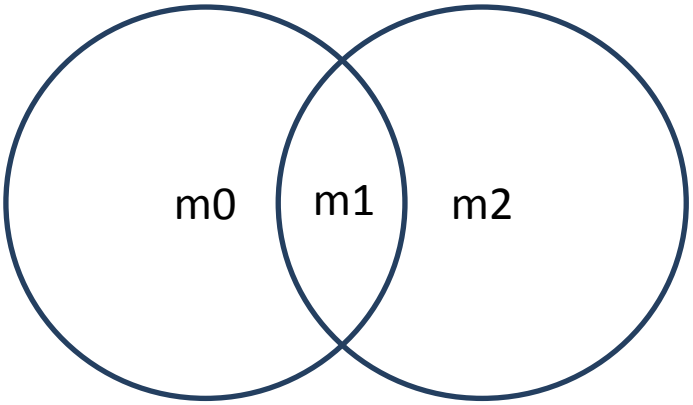
Since this backtracking and forward checking algorithm is simple enough to implement, it is

ideal for online real time missile assignment with a small response time. The detailed source code for this algorithm can be found in the appendix.

7.7 Communication and Physical Limitation

The above design assumes a perfect situation. However, in reality, imperfect situations do happen. Imperfect situations include different noise levels, more complicated transfer functions, computer hardware limitations, imperfect communications, etc. Here we discuss two imperfect situations, imperfect communications, and computer hardware limitations. The rest of the situations might be dealt with in future work.

Imperfect communication can be considered in two parts. The first one is that not all missiles are within the same radio range. That is, there might exist a situation where one missile in the group might not be able to receive the messages sent from another missile because the distance between the two missiles is beyond the radio range. This situation can be described by the following figure:



As the figure shows, m0 and m1 share the same radio region, which means m0 and m1 can send and receive messages from each other, the same for m1 and m2. But m0 and m2 cannot reach each other due to the too long distance between them. If this is the case, at the first time step, m0 gets messages from m1, and sends messages to m1; m1 gets messages from both m0 and m2, and sends messages to both of them; and m2 gets messages from m1 and send message to m1. At the second time step, m0 still gets messages from m1, but now since m1 already has the messages from m2,

therefore m0 also can get messages from m2, and so can m2. This is called message forwarding. The on board computer in each missile should wait until each missile receives the same information table, in order to avoid the situation that multiple missiles are assigned to the same target or vice versa.

The second case may arise when one of the missiles has a broken radio during online flight time. In this case, other missiles should not wait forever for this broken radio. Thus, there must be some timeout mechanism.

In order to deal with imperfect communications, our algorithm should be modified accordingly, as follows:

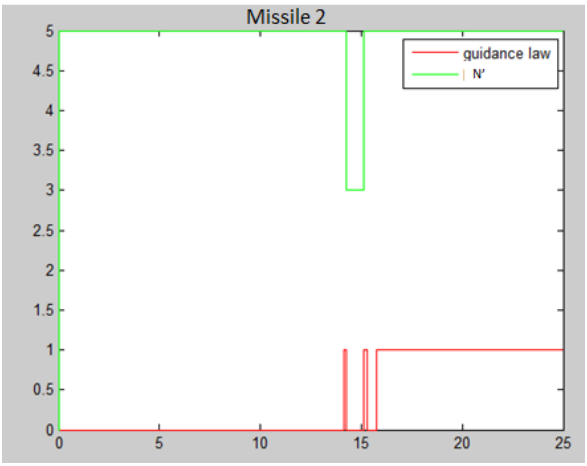
Miss_Info[][] table Start with all -1s for each member.

Continuingly update with 0s and 1s from other members. If receiving a -1 from another missile but the same entry of the information table already has a 0 or 1, ignore the -1.

If there is no update for two periods, send ready to switch signal

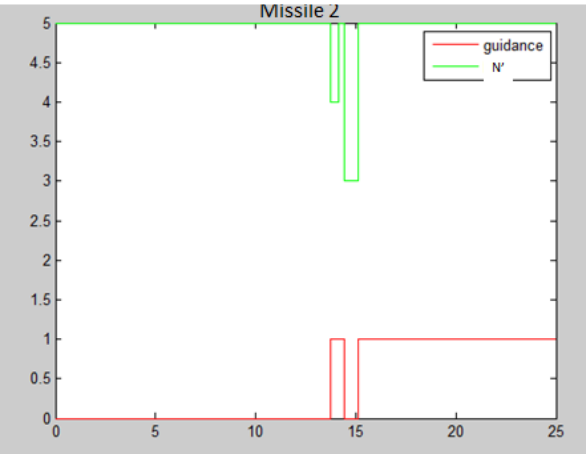
Replace remaining -1s with 0s.

As for the mechanical hardware limitation, sometime it might be that the command output for the algorithm cannot be changed very immediately. For example, if ignoring hardware limitations, following situation might happen:



From the figure, the online guidance law for missile 2 was changed very shortly during around 14s to 14.01s, which in reality might not happen due to the hardware limitation. Therefore, after

modifying the Simulink model for the hardware, following result occurs:



Notice that even if there is a hardware limitation, the system can still performs successfully and all missiles would successfully hit the targets, as we will see in the next section.

7.8 Simulation Results

In order to demonstrate the advantage of the OCAA, it is helpful to compare it with traditional PNG. Below are a couple of results showing 3 missiles and 3 targets engagement system.

Assumptions and initializations:

No Noise

All targets are mass points with transfer function$\frac{1}{0.5s+1}$; Missile is a rigid body, influenced by aerodynamic forces (has velocity reduction); and Missile Airframe is 1st order system, time constant of 1, with a transfer function: $\frac{1}{s+1}$

Target 1 sensing range: 800m; Target 1 maneuvering level: 15g;

Target 2 sensing range: 1200m ; Target 2 maneuvering level: 20g;

Target 3 sensing range: 800m ; Target 3 maneuvering level: 10g;

Missile maneuvering saturation: 3 times as much as target maneuvering. Same missile model.

All Initial targets velocity: 350m/s [6]; all missiles Initial Velocity: 700 m/s [6] (air to air supersonic missiles).

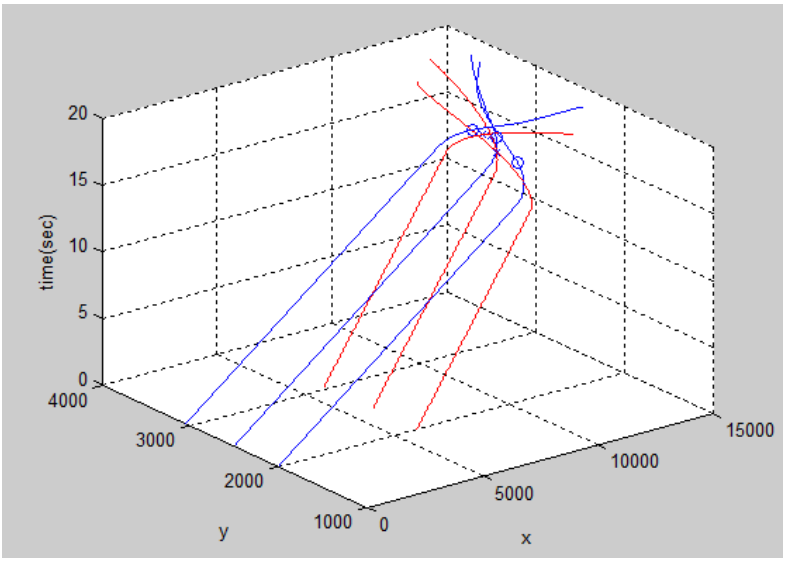


Figure 7.7: Simulation results for PNG. The three blue lines are missile trajectories, the red lines are target trajectories. Once the missile enters its explosive range, it explodes, and this missile and its target stop moving. This figure shows that all of the missiles and targets never stop moving, which means none of the missiles ever enters their explosive range. The detailed final miss distance for each missile target pair is shown in the table below.

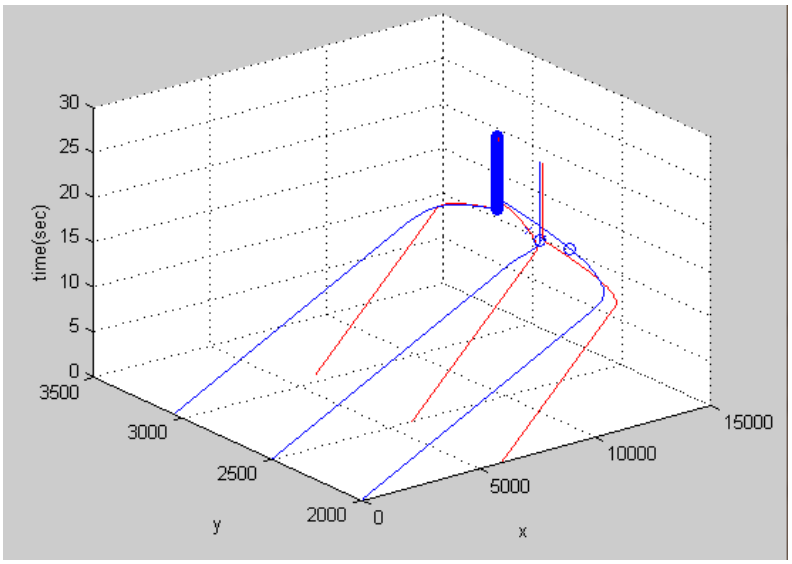


Figure 7.8: Simulation results for OCAA showing that around 17s to 18s all three missile target pairs stop moving, which means that all three missile successfully enter their explosive range and blow off all three targets.

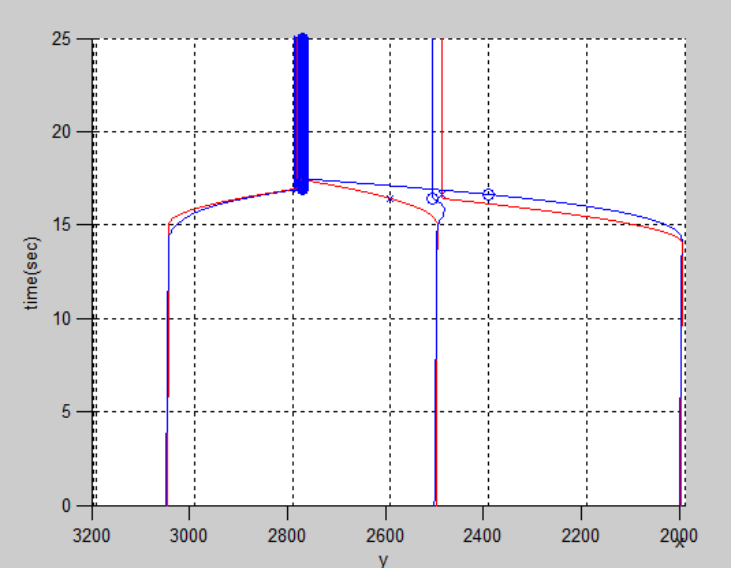


Figure 7.9: Same figure as last one, but with different angle: time vs y axis.

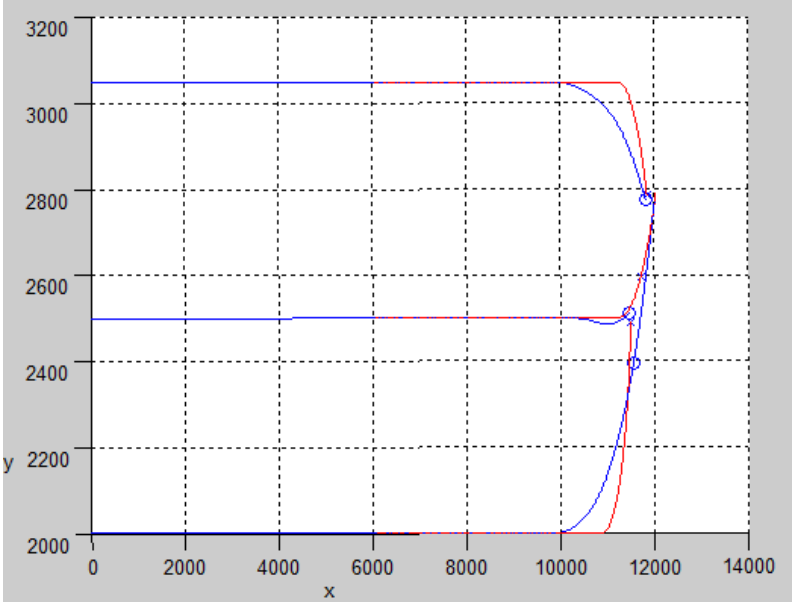


Figure 7.10: Same figure as last one, but with different angle: x vs y axis.

	Target 0	Target 1	Target 2
Miss Distance for PNG	118m	225m	55m
Miss Distance for OCAA	29m	29m	29m

Table 7.1: Miss Distance comparison between two algorithms

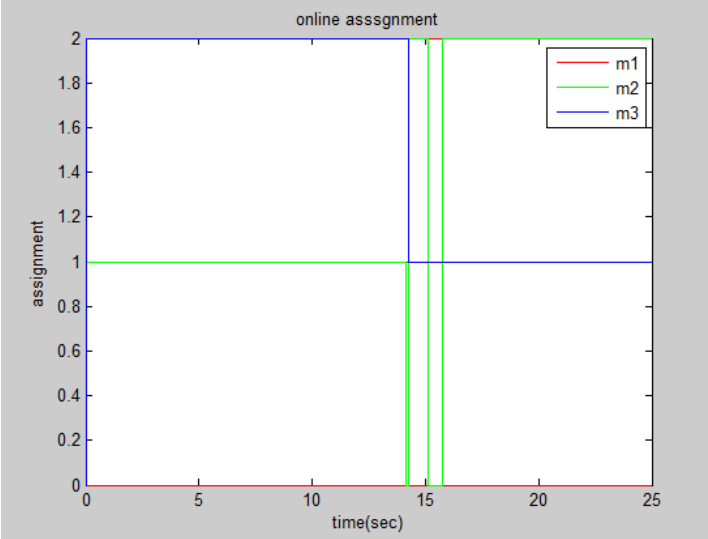


Figure 7.11: This figure shows the detailed online assignment adjustment for each missile member

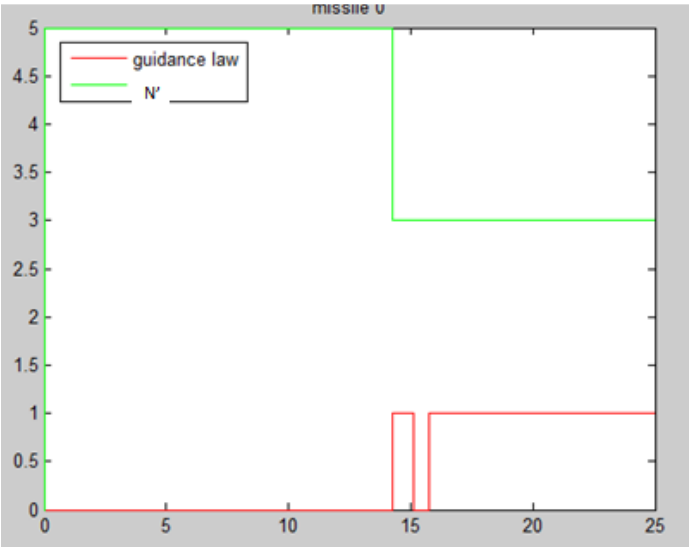


Figure 7.12: This figure shows the online adjustment for guidance law and guidance parameter N' for missile 0.

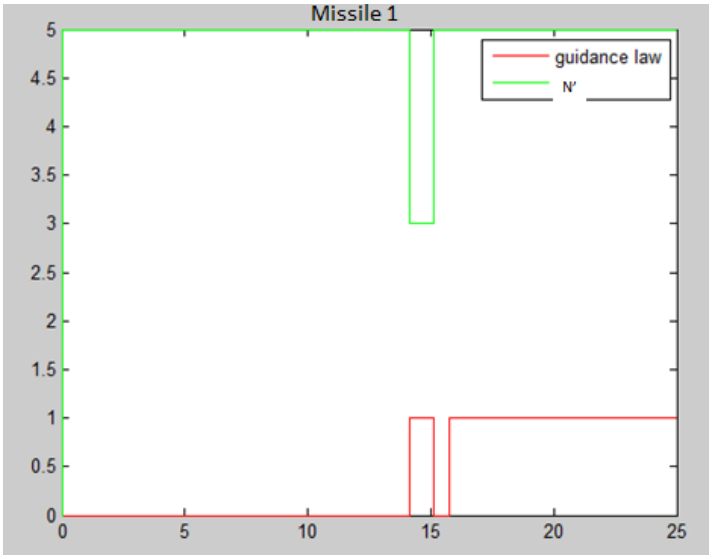


Figure 7.13: This figure shows the online adjustment for guidance law and guidance parameter N' for missile 1.

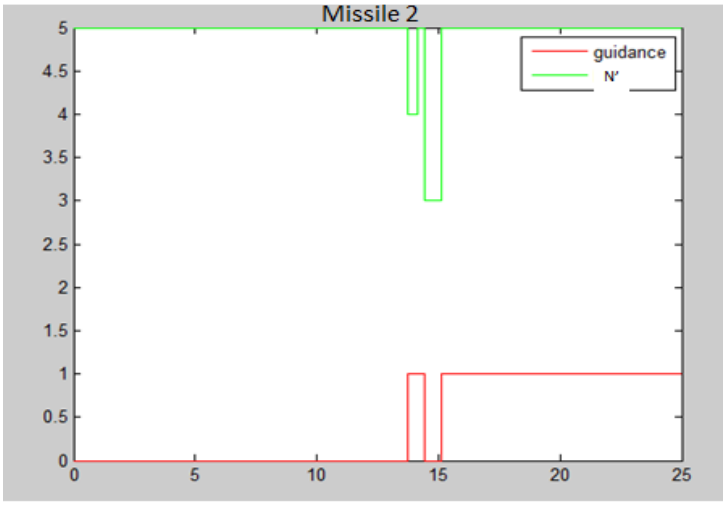


Figure 7.14: This figure shows the online adjustment for guidance law and guidance parameter N' for missile 2.

Flight Time	Missile Online Assignment
0 - 14s	[0, 1, 2]
14 - 15s	[0, 2, 1]
15 - 16s	[2, 0, 1]
15 - final	[0, 2, 1]

Table 7.2: Intermediate Online Adjustment.

Computer response time (seconds)	Final result for all targets
0.01	all hits
0.02	all hits
0.03	all hits
0.04	all hits
0.05	2 hits, 1 miss
0.1	2 hits, 1 miss
0.2	2 hits, 1 miss
0.3	2 hit, 1 misses
0.4	1 hit, 2 misses
0.5	three misses

Table 7.3: Performance in terms of computer response time; "hit" ("miss") means that the final miss distance is within (outside) the missile explosion range.

As we can see in Figures 7.7 to 7.10 a multiple-missile multiple-target system using communication and online cooperation, could achieve a better performance than a single-missile single-target system where each missile employs its initial algorithm without online cooperation. Moreover, Table 7.3 also shows that a multiple missile and target system can handle the problem of computational delay well. All three missiles are within their explosion range (of 30 meters) even for a delay time of 0.05s. Also, not all targets are missed until the computer response time is as large as 0.5 seconds, an unlikely case for modern computers.

CHAPTER 8

FUTURE RESEARCH

The computer response time for the missile guidance and other computer algorithms in the missile airframe is generally ignored in previous papers in the open literature. This thesis studies the impact of computational delay for both single-missile single-target system and multiple-missile multiple -target system.

Future work may include a more realistic missile and target model, noise in the sensor, a more sophisticated online cooperation algorithm, and additional physical limitations. For the latter, this paper uses a simplified missile and target model, assuming that the actuators are perfect without internal mechanical delay, and assuming that the missile and target airframe model are first order systems. A more realistic model could be used for more accurate results.

APPENDIX

THESIS SOFTWARE TOOLS MANUAL

Introduction

The tools used in this thesis are MATLAB and Simulink, Truetime Realtime Kernel from Lund University, and C++ Visual Studio 2008. In the tools package, “engage.mdl” is a Simulink model for the entire simulation. “online.cpp” is the C++ source code for all the algorithms for the online missile engagement, including Missile Guidance Laws (PNG and APNG), and Online Cooperative Adjustment Algorithm (OCAA). The source code is embedded in a real time kernel block called Truetime Real Time Kernel developed by Lund University. “Init.m” is the initialization for missiles and targets, written in MATLAB. “plot_engage.m” is the plot file used to plot simulation results. Before running the Simulink simulation, make sure that the Truetime Kernel is appropriately set up. One can refer to: http://www3.control.lth.se/truetime/ for the software setup and other manuals.

Run engage.mdl simulation model

After we have set up all the required software and library, we can actually run the engage.mdl simulation model by following matlab commands:

- Start matlab. Go to the simulation directory
- >>make_truetime
- If any of the online.cpp file is changed: >>ttmex –g online.cpp
- Run init.m file
- Run engage.mdl file
- Run plot_engage.m file to see the simulation results.

REFERENCES

[1] Shin, K.G., and Cui, X., 1995, "Computational Time Delay and Its Effects on Real-Time Control Systems," *IEEE Transactions on Control System Technology*, Vol. 3, No. 2, pp. 218 - 224.

[2] Nilsson, J. Bernhardsson, B., and Wittenmark, B., 1998, "Stochastic Analysis and Control of Real-time Systems with Random Time Delays," *Automatica*, Vol. 34, No. 1, pp. 57 – 64.

[3] Junmin Wang, Raul G. Longoria, 2006, "Effect of Computational Delay on the Performance of a Hybrid Adaptive Cruise Control System," Vehicle Dynamics and Simulation 2006 (SP-2018).

[4] F. William Nesline Jr. and Paul Zarchairf, 1985, "Digital Homing Guidance—Stability vs Performance Tradeoffs," J. Guidance, Control and Dynamics, 1985.

[5] George M. Siouris, 2004, "Missile Guidance and Control Systems," Springer; March 2004.

[6] P. Zarchan, 2008, "Tactical and Strategic Missile Guidance," AIAA; 5th edition, January 2007.

[7] M. Guelman, 1972, "Proportional navigation with a maneuvering target," IEEE Transactions on Aerospace and Electronic Systems, No. 3, May 1972.

[8] M. Guelman, 1976, "The Closed-Form Solution of True Proportional Navigation," IEEE Transactions On Aerospace And Electronic Systems, Vol. Aes-12, No.4, July 1976.

[9] Uday S. Shukla, Pravas R. Mahapatra,1990, "The Proportional Navigation Dilemma-Pure or True?" IEEE Transactions on Aerospace and Electronic Systems, Vol. 26, No. 2, March 1990.

[10] Paul B. Jackson, 2010, "Overview of Missile Flight Control Systems," John Hopkins APL Technical Digest, Vol.29, No. 1, 2010.

[11] Lund University, Sweden, "Truetime Real Time Kernel".